Editors:
S. Strzelczak, P. Balda, M. Garetti, A. Lobov

Open Knowledge-Driven Manufacturing & Logistics The eScop Approach

Editors:
S. Strzelczak, P. Balda,
M. Garetti, A. Lobov

Open Knowledge-Driven
Manufacturing & Logistics
The eScop Approach

ISBN: 987-83-7814-440-3

Warsaw 2015

OPEN KNOWLEDGE-DRIVEN MANUFACTURING & LOGISTICS THE ESCOP APPROACH

Warsaw 2015

External Reviewer:
Prof. Leszek Pacholski

Open Knowledge-Driven Manufacturing and Logistics - The eScop Approach

Edited by Stanisław Strzelczak, Pavel Balda, Marco Garetti and Andrei Lobov

ISBN 978-83-7814-440-3

CONTENTS

Preface V

Foreword IX

PART ONE – Towards Open Knowledge-Driven Manufacturing and Logistics - Thriving on Challenges and Opportunities

Industrial Production - Type of Processes, Current Needs and Emerging Challenges .. 3
Garetti, M., Fumagalli, L.

Ontology-Aided Manufacturing and Logistics 23
Strzelczak, S.

Designing of Business Models for ICT Products 51
Wojtachnik, R.

Business Model Visualisation for ICT Product 67
Wojtachnik, R.

Business Models for Open Knowledge-Driven Manufacturing Execution System 85
Räsänen, S., Lederbuch, P.

Holistic System of Evaluation of Production Systems Efficiency 95
Marciniak, S.

PART TWO – Implementing Ontologies

Implementing Ontologies in Manufacturing and Logistics - From Theoretical Fundamentals to Prospects 111
Strzelczak, S.

Ontology-Based Modeling of Production and Logistics Systems 215
Garetti, M., Negri, E., Fumagalli, L.

Classification of Knowledge Representation Implementations in the Manufacturing Systems Domain 235
Ramis Ferrer, B.

Methodology to Develop Ontology for Manufacturing Systems 245
Sampath Kumar, A.

Ontology-Based Backend Engine for Manufacturing and Logistics Systems 259
Xu, X.

Ontology-Driven Visualization for Manufacturing Systems 273
Sampath Kumar, A.

PART THREE – Towards (Web-)Service Oriented MES Architectures

State-of-art of Manufacturing Execution System - A Technology Review 283
Xu, X., Räsänen, S., Camp R.S.

Developing Open Knowledge-Driven Manufacturing Execution System 295
Iarovyi, S., Xu, X.

On Service Composition - Dynamic Formation and Orchestration of Service Workflows 311
Lobov, A.

Knowledge-Based Production Planning - Identification of Model Structures for Production Planning Procedures 319
Borowiecki, T., Stryjski, R.

A Proposal of Decentralized Architecture for OKD-MES 331
Ramis Ferrer, B.

REST-Enabled Physical Devices in Service Oriented Architecture 341
Severa, O., Pišl, R.

Assessment of Communication Protocols for Industrial Devices 355
Gonzalez Moctezuma, L.E.

eScop Project Physical Layer Development - An INCAS Mini-pilot Case Study 369
Balda, P., Štětina, M.

Cyber Security for Web Service-Based Industrial Devices 383
Gonzalez Moctezuma, L.E.

About Authors.....399

Preface

European economy, and in particular European industries, are recently exposed to many challenges. Diversified and fragile markets provide generic uncertainty and variability, while globalization of industrial networks enhances openness and complexity of operations. Consequently changeability and reconfigurability of production systems and processes are of quickly growing importance. With regard to this specific issue, the book aims to respond to the economic and social challenges of European development. It is by a provision of specific conceptual and technical developments, which focus on the openness and changeability of manufacturing and logistics systems, especially at the shop floor level. It is argued herein that the both requested characteristics can be facilitated by a combination of particular technologies, namely:

- Ontology-based representation applied for knowledge based operation of systems and processes,
- Embedded systems or appliances,
- Service oriented architectures.

This approach was proposed within the application for the EU JU ARTEMIS eScop project: *Embedded systems Service-based Control for Open manufacturing and Process automation*. Later on it is also referred as the eScop approach.

The book brings together the opinions of experienced scholars, young researchers and practitioners, who all together attempted to comply with the emerging needs and requirements. They tried to find an answer to the question about how to merge the founding technologies to provide a basis for sound and robust solutions. The range of key topics addressed in the book includes:

- Manufacturing Execution Systems (MES): functionalities, development, design patterns,
- Methodologies and architectures for Open Knowledge-Driven Manufacturing Execution Systems (OKD-MES),
- Knowledge representation and ontology-based modeling and management for the industrial domain,
- Service-driven operations and using Web services to support them,
- Future needs and requirements, with regard to possible extensions of the eScop approach.

Through individual chapters, authors present views, concepts, approaches, and developments around the key topics. Hence the readers should have learnt the major issues concerning the eScop approach, being currently addressed by the research, or being experienced by practice and developments. The book is composed of three parts, each of them focusing on specific areas.

Part One provides multiple highlights concerning various needs, requirements, opportunities and challenges, to be addressed by the new OKD solutions, particularly in reference to the OKD-MES. A distinctive attention is given to the business models

that could eventually facilitate a widespread dissemination of such novel developments throughout industries. The first chapter (by M. Garetti and L. Fumagalli) investigates a range of determinants that provide different insights for development of OKD-MES solutions, including: the specific characteristics of discrete and process industries, and with this regard conclusions from the former developments of enterprise architectures; predictions about the needs of future industries derived from some roadmaps prepared by EU and EFFRA, from opportunities offered by the novel ICT and automation technologies, and consequently from the requirements and performances expected from the future OKD-MES. The opening chapter finalizes with an introduction of the new paradigm of open automation manufacturing. The next chapter (by S. Strzelczak) continues the discussion of groundings for future OKD solutions. It provides an extended perspective for various OKD implementations in the area of manufacturing and logistics, while focusing on the potential advantages of ontology engineering, mostly with regard to the developments built upon capacities of the Semantic Web. It is argued that the eScop approach can be generalized, therefore more widely adopted in the domain of interest, like in some other areas. A number of systemic and functional ideations is presented with this regard. The highlights given herein are supported by conclusions from the research of industrial needs and requirements performed along the eScop project, as well as by a literature review. The two following chapters (both by R. Wojtachnik) introduce in-depth the concepts of business models. A lot of attention is given to practical aspects of successful introduction of ICT developments to the markets. As the open solutions often provide a particular challenge with this regard, many success stories from the past are commented. The fifth chapter (by S. Räsänen and P. Lederbuch) continues investigation of the business models methodology, however focusing on the OKD-MES solutions. The presented analysis of business models for OKD-MES is supported by some results of research performed by the eScop project team. The final chapter of Part One (by S. Marciniak) proposes a framework for holistic assessment of production systems efficiency. The methodology respects the paradigms of the new economy, including the sustainability paradigm, all being promoted by the EU, particularly along assessment of the EU supported R&D programs and projects. The framework goes beyond the existing methodologies, which commonly recall the developments concerning flexible automation.

Part Two is centered around the major problems that arise along design and implementation of OKD solutions in the manufacturing and logistics domain by using ontologies. The introductory chapter (by S. Strzelczak) reflects on the provisions by ontology engineering, focusing on those developed by the World Wide Web Consortium (W3C), which were adapted within the founding concepts of eScop approach. Specific complexities are discussed, which occur along manufacturing and logistical operations, and exhibit dynamic and mereotopological nature. It is argued that they may lead to particular requirements that are qualitatively different from those presumed by the concepts of Semantic Web. Therefore theoretical foundations of ontologies are reconsidered and a new transformational framework is introduced, which is more generic and expressive than the existing ones. It is built upon a dyadic construct of the operational pair ‘resources-tasks’. The next chapter of part two (by M. Garetti

et al.) discusses the requirements for ontology-aided modelling of manufacturing systems. Then, it continues with a proposal of an extended ontology for manufacturing systems domain. The following chapter (by B. Ramis Ferrer) reflects on variety of knowledge representation implementations, especially in the manufacturing automation domain. A relevant classification is proposed to differentiate the requirements that have to be fulfilled by the identified types of knowledge representation. This way a reference aid is obtained, to support the design of OKD solutions. Followingly, a methodology to develop ontology is presented (by A. Sampath Kumar). It addresses representation of the manufacturing systems in an ontology format, and defines the steps, techniques and tools involved in the process of ontology development. The proposed methodology is expected to be more efficient when compared with the common approaches. The next chapter (by X. Xu) investigates functionalization and operationalization of ontologies, with a special regard given to the OKD-MES solutions. A relevant architecture for the ontology-based backup engine is presented, which is based on integration of the Fuseki server to support execution of the CRUD (Create, Read, Update and Delete) operations on ontologies. The last chapter of Part Two (by A. Sampath Kumar) proposes a representation of the knowledge required for dynamic visualization of manufacturing systems by the means of ontology, to assure a homogenous representation of all knowledge and data about any system.

Part Three discusses major developments that are expected to found basic principles for MES systems design, which are principally based upon Service Oriented Architectures and Web Services. The first chapter (by X. Xu et al.) reviews State-of-Art of the Manufacturing Execution Systems, taking the technology perspective. It reflects on evolution, available market solutions, future trends and functionality related standards, this way providing insights for the OKD-MES developments. The following chapter (by S. Iarovyii and X. Xu) explores the OKD approach used for the development of MES systems. It attempts to exploit a synergy of the Service-Oriented Architecture, following the concept of Web Services, and embedded devices. Therefore, an in-depth review of the architecture that respects the eScop approach to OKD-MES is presented. Followingly, the consecutive chapter (by A. Lobov) discusses the orchestration and composition of services. Taking the eScop approach as a reference point, an approach to orchestration is described, which allows dynamic formation of service hierarchies in accordance to current production needs. The approach enables keeping track on all the existing service workflows. Thus, the locus of control for the overall system is also kept. The next chapter (by T. Borowiecki and R. Stryjski) investigates theoretical foundations for knowledge based production planning and control (PPC), according to the Constraints Programming (CP) paradigm. Modelling of the PPC procedures by controlled search process is proposed, using the means of the CSP/COP (Constraints Satisfaction Problem / Constraints Optimization Problem) formalization for declarative descriptions. The architectural considerations are continued in the next chapter (by B. Ramis Ferrer). A reduction of vertical communication within the MES systems is targeted, basing on a decentralized architecture, which exploits downward delegation of the OKD-MES functionalities to the level of embedded devices, which in particular are expected to host ontologies. Subsequently, the consecutive chapter (by O. Severa and R. Pišl) explores the capacities of embedded

devices, which are enabled by application of the REST protocols. A major attention is given to advanced Web Service management and Web Service interfacing. The following contribution (by L.E. Gonzalez Moctezuma) continues the topic, and compares in a structured way alternative communication protocols, which are commonly used for industrial communication at the level of devices. Therefore helpful recommendations are defined for the OKD-MES development. The next chapter (by P. Balda and M. Štětina) provides conclusions that were derived along a pilot development in the eScop project. A network of configurable embedded devices, communicating by the RESTful Web Service interface, was installed at the premises of the INCAS company in Vigliano Biellese. An overall development methodology has been derived from the gathered experience, which encompasses simulation, and uses the model-based design technique (software in the loop). The last chapter of the book (by L.E. Gonzalez Moctezuma) discusses the security related problems of Web-Service based industrial devices. The different security threads identified in industrial systems are analyzed, aiming at the possibility to implement security mechanisms in devices that implement the Web Services. With this regard, a set of decision diagrams is presented to support selection of the architectural components and security protocols, depending on the system requirements, characteristics and context.

As the editors, we would like to thank all of the contributors for preparing their chapters to a high standard. A warm thanksgiving is due to Prof. R. Bacewicz, the Vice-Rector for Research of Warsaw University of Technology (WUT), and to Prof. S. Marciniak, the Head of Department for Manufacturing Systems Organization at WUT, who kindly promoted publication of this book. Each of the chapters was twice peer reviewed. This included external reviewing by Prof. L. Pacholski, to whom a special gratitude should be addressed for his helpful comments. Last but not least, a separate word of thanks should be given to Ms. B. Makuch, who put a lot of effort to assure a smooth preparation of the publication.

The research leading to most of the results presented in the book has received funding from the EU ARTEMIS JU under grant no. 332946 correspondent to the project eScop (Embedded systems for service-based control of open manufacturing and process automation). The partners also express their gratitude for support to respective national funding authorities. The Czech partners thank to the Ministry of Education, Youth and Sports; the Finnish partners to the Finnish Funding Agency for Technology and Innovation; the Italian partners to the Ministry of University and Research, and the Polish partners to the National Centre for Research and Development.

Stanisław Strzelczak
Warsaw University of Technology, Poland

Pavel Balda
University of West Bohemia, Czech Republic

Marco Garetti
Politecnico di Milano, Italy

Andrei Lobov
Tampere University of Technology, Finland

Foreword

The social and economic life in recent decades was undergoing many revolutionary changes. Exemplifications can be illustrated in different areas, ranging from social communication, commerce, banking, through smart products, buildings and cities, to advanced systems of weapons. Most of such developments were driven by innovations in information and communication technologies (ICT) and automation technologies (AT), like: the Internet, Big Data, Semantic Web, cloud computing, knowledge-based systems, embedded devices, smart appliances, tracking technologies et al. The collective term Industry 4.0 is often used to couple those of them, which are expected to bring crucial opportunities for future industrial development. Consequently, advancements in areas like Cyber-Physical Systems, the Internet of Things, Smart Grids et al., are being intensively promoted by governments and international organizations, while getting high priorities in research agendas of universities and companies.

The industrial domain, and in particular the manufacturing and logistics, are among those areas, that are significantly influenced by the forenamed trends. Furthermore, the manufacturing industries are in the meantime affected by various external challenges and exposures. To name just a few of them, which seem to be most important: fragmented and fragile markets; diversified demand; shortening life-cycles of products, systems, and technologies; increasing networking and spatial spread of operations. All they bring about the rapid growth of openness, variability and complexity, while simultaneously calling for enhanced changeability. Changeability has become an important asset for production activities, especially in the developed economies, like in Europe. The concept of changeability can be variously instantiated, e.g. by flexibility, agility, modularity, re-configurability, transformability etc. The presented book focuses on the re-configurability and transformability of manufacturing and logistical systems. Both concepts refer to structural changes, which are fundamental to production resources and processes. This way the book aims to respond to one of the most important challenges, being recently faced by European economy, and in particular by European industries. With this regard it is understandable why the EU JU ARTEMIS eScop project, which provides a major source of contributions to the book, has got the highest rank among the applications.

The book argues that significant improvements in re-configurability and transformability of production systems can be facilitated by merging the power of particular technologies. While embedded systems and Service-Oriented Architecture (SOA) provide a grounding base for the openness and knowledge based operations, the representation of systems' complexity in terms of ontology engineering makes easy re-configurability and transformability possible. It is notable, that such an ideation is of much more general use, than just for the case of shop floor control level of production operations and Manufacturing Execution Systems. It can eventually support other applications for different domains, assuming that changeability, as referred to resources and processes performed by systems mixing human and technical components, is crucial for their operation. Actually, some comments about this kind of extendability of the proposed approach, are given in several chapters of the book.

The theme of the book is presented by a comprehensive collection of chapters, targeting different aspects, from theoretical through technological up to implementation related considerations. This way its reader can gather a holistic picture of the problem domain. Alternatively, she or he can focus some particular issues, as presented in respective chapters. It is an advantage of this book, in opposition to many other publications, which usually tend to address only some selected aspects of the discussed topic. I believe that this quality will help to gather a big audience for the book, and to enhance its impact.

Finally, I would like to express my pleasure, that such an important contribution, providing a good response to significant R&D challenges, could be supported by the Warsaw University of Technology and its Publishing House.

Prof. Rajmund Bacewicz

Vice-Rector for Research
Warsaw University of Technology, Poland

PART ONE

TOWARDS OPEN KNOWLEDGE-DRIVEN MANUFACTURING AND LOGISTICS – THRIVING ON CHALLENGES AND OPPORTUNITIES

Industrial Production

Type of Processes, Current Needs and Emerging Challenges

Marco Garetti and Luca Fumagalli

Politecnico di Milano, Milano, Italy

{marco.garetti, luca1.fumagalli}@polimi.it

1 Introduction

Industrial production is one of the main pillars for assuring human welfare, basing more and more on advanced technology for continuous improvement of products and related production processes. In this chapter, after this introduction, section 2 is dedicated to the distinction of production processes between their two fundamental types, which are process production and discrete production. They are explained in section 4 and 5 respectively, in reference to different characteristics they have, both from the physical and the control point of view. To this regard, section 6 is dedicated to a review of the enterprise modelling approaches and the related standards that have been developed in the past years for the industrial production domain. Then, section 7 introduces the topic of how the current worldwide competition creates specific needs for industrial production in developed countries. In section 8, it is analyzed how new ICT solutions and advanced automation solutions can provide opportunities for improvement, which are analyzed separately as market conditions, related production performance objectives and requirements in sections 9, 10 and 11 for the discrete manufacturing sector, and in sections 12, 13 and 14 for the process production. Finally, referring to the eScop research project, section 15 illustrates the new paradigm of open automation manufacturing, which is postulated to satisfy to the most part of the previous requirements and to overcome the main limits of current solutions.

2 Industrial production types

It is possible to classify industrial production in two main categories: process production and discrete manufacturing, which are implemented into two different types of production systems. It is important to underline that some literature sources refer to "process production" as "process manufacturing".

2.1 Process production

Typical feature of process production is that the components of which the final product is made can no longer be distinguished at process completion: the product cannot be decomposed backwards, as the original components are no longer distinguishable

or have changed in their nature. Examples of this production type are the processes used to obtain steel, paper, cement, chemicals, glass, pharmaceuticals, and etcetera. The production system is characterized by a well-defined technology cycle (often the production process does nothing but to reproduce the appropriate conditions to make a series of physical and chemical transformations, as happens, for example, in the case of steel). The output of this type of production (indeed called primary) is often the raw material (like a steel bar) acting as the input for the following discrete manufacturing processes (as for example gears obtained from machining separate segments of the steel bar). As such, large amount of product quantities (e.g. tons of steel bars) are required for this type of productions, therefore process production plants are generally large scale production system, also because they can benefit from the economy of scale principle (i.e. larger plants / equipment are more cost effective than smaller ones). Peculiarity of these processes is the in-line arrangement of the process workstations, due to the fixed nature of the technology cycle, they have to perform, and thus the process machines can be conveniently arranged in sequence (line). The continuous flow of the raw material and the standard type of product are the essential features of these systems for achieving significant economies of scale. In this type of process technology, the production system can often look like a single large "machine", that is to say the result of a unitary technical concept in which the chemical and physical transformations take place on an industrial scale size. Nonetheless, in some cases (production of different products of low volume), the discontinuous flow batch type of production system can be used as a configuration for process production. In this case, the layout of the production system is not arranged in a line, but in departments in which similar machines are grouped.

2.2 Discrete manufacturing production

In this type of production products are made by a number of discrete components, or parts, generally of a different nature (examples are products such as cars, home appliances, electronic equipment, shoes, toys, etc.). A typical characteristic of this type of productions is that a product can be assembled and disassembled (this condition may not occur, but without losing the general meaning, as in the case of assembly by welding, gluing, forcing, sewing of the components). In principle, the production process includes two steps: the phase of manufacture (fabrication), to be understood as a set of processes (normally made by machines) that modify the shape, the dimensions or the surface status of individual parts; and the phase of assembly, to be understood as a complex of operations of juxtaposition of individual parts (made by machines or human operators) to form an assembly. Discrete production is characterized by a great variety in the technology cycles (routings) for fabricating and assembling components, parts and final products (these cycles often admit several variants).

The process is realized by passing a variety of different machines, which can carry out several working cycles (e.g. flexible numerical controlled machines) and thus produce different products at different times. Each product is manufactured following a working cycle through the machines (called routing). The loading and unloading of machinery occurs at time intervals and during these phases the system is idle.

Production engineering plays an important role in this type of production being in charge of designing the process routing for manufacturing the product's parts in manufacturing shops or lines, and then to assemble the parts in assembly shops or lines.

2.3 Type of process equipment

In addition to the above conceptual differences between process production and discrete manufacturing, one main important point is that a different type of equipment is used in these two industrial contexts:

- In process production, the raw and in-process material are normally made by fluids (liquids / gasses) or bulk solids that flow along the process. Therefore, process equipment is made by tanks, reactors, pipes, pumps, heat exchangers and so on.
- In discrete manufacturing, the raw and the inbound process material is made by discrete parts of different materials, which, for example, are machined on machine tools and then assembled for obtaining the final product.

Because of all these substantial differences, it is also worth considering that different production management policies are applied to the two industrial contexts (this aspect is further detailed in the following sections).

More details on the equipment solutions used in process production and in discrete manufacturing are reported in the following paragraph, in which the characteristics of the solutions in the two industrial domains are described also considering the issues to be addressed in modelling the domains.

3 Configurations of process production systems

Process production plants can be characterised by continuous or discontinuous flow systems (continuous and discontinuous flow systems correspond to different design, management and control approaches). For what concerns material flow, raw materials are converted into finished products through a series of production stages involving chemical and physical transformations, while semi-finished products may be stored within the process, but only in specific stages of production, since work-in-progress sometimes cannot be stored. In any case, this possibility depends on the specific plant layout. Raw materials and semi-finished products may be split to provide more specialized processing and they may be combined to create finished products.

4 Configurations of discrete manufacturing systems

Discrete manufacturing plants can be characterised by job-shop configuration, cell-manufacturing configuration or line-manufacturing configuration (job-shop, cell and line configurations correspond to different design, management and control approaches). For what concerns manufacture, the production system is composed by machines that operate on singularly separated items. For what concerns assembly, the produc-

tion system is composed by assembly stations (manual or automatic) that can be organized as assembly shops or assembly lines. For what concerns material flow, raw materials are converted into finished products mainly through mechanical operations (also including assembly operations), while semi-finished products may be stored in almost any phase of the process; of course depending on the production system layout and presence of buffers (this also creates more flexibility in the production management). Any single item can be personalized through the specific operations applied to it. This strongly depends on the flexibility of control on the single machines of the production system. The higher is the customization applied to the product, the higher is the required capability to flexibly manage the production and thus to control the machines. Finally, items may move along the plant separately or can be grouped in larger movable units (i.e. pallets, lots, carts).

Operations can be carried out by fully automated systems (typical for large plants with high volumes of production or for flexible manufacturing cells with middle/low volume of production) or can be supported by operators, also with mixed situations. In this case, the production management and control tools must also consider operators' activity information and not only machines' information to properly manage the production.

5 Management and control of production systems

Management and control of production systems is the activity that allows managing the resources of a production system with the objectives of satisfying the customer demand, while minimizing the production cost and respecting all constraints related to quality, safety and human and environmental protection.

Given these common objectives, differences stands between the control requirements of process production and discrete manufacturing. The main difference is related to the control of the machines with the flow of material and the synchronization among them. Indeed, raw materials must be arranged in order to satisfy customer demands that may require the production of different types of final products, requiring management of the production activity, however:

- In process industry, this mainly results in the management of the process parameters to be set to apply the required transformation on the flow of material that is generally continuous. This may somehow simplify the routing and the management of the internal logistics within the production system, but requires an accurate monitoring and control of the machine parameters. Another very important aspect is that the nature of the raw material (i.e. fluid or bulk) and the type of process (with dedicated systems) allows to suppose that, both the material being processed and the related equipment, are permanently ready and available for the process activities. This allows formulating the control of the manufacturing activity with a near real-time approach. Therefore, the problem of supply lead-time only occurs at the high level of material acquisition from outside the factory, where appropriate control must be structured in levels with different time horizons and different time frames (time buckets). Consequently, the manufacturing execution system (MES)

in process industry (which is also called process control system) can cover many of the phases of the in-house production activity. Instead, the production planning activities, related to the supply of materials and the warehouse management for raw material and finished goods, are managed in an asynchronous way.

– In the discrete manufacturing the situation is substantially different, because the product is made of parts, which are partially produced in house, and partially outside the factory. Furthermore, parts are produced individually on shared machines, thus requiring conflict management of access to machines. Moreover, the sharing of resources (machines, assembly stations, etc.) among products and the presence of set-up times generate the need of lot production, therefore routing and management of lots of production becomes more crucial. Indeed, the possibility to clearly separate a single item and singularly manage its production, the multiplicity of parts composing the final product and the lead-times affecting the different stages of production, make very complex the management phase of production. In particular, it is necessary to introduce various levels of planning and scheduling which must be able to make a reconciliation in an asynchronous way of the production flows (Fig. 1). As a consequence, the manufacturing execution system (MES) in discrete manufacturing stands below the production management activity, which in turn is divided in many phases with different time horizons (i.e. in Fig. 1: master

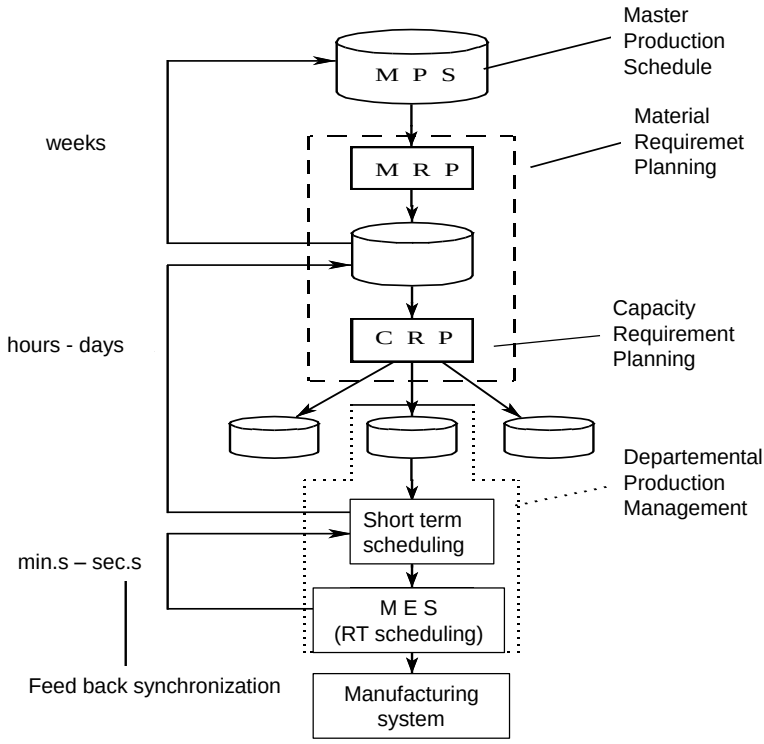


Fig. 1. Simplified production management architecture in discrete manufacturing

production schedule, material requirements planning, capacity requirements planning, short-term scheduling and real-time scheduling within the MES). The real-time production scheduling activity represents the link between the upper levels of production planning and the real time activity of the MES.

6 Enterprise modelling and related standards for the manufacturing domain

Enterprise modeling is an important foundation for a clear design of the management structure of production systems; therefore, a short review of its development is presented in this paragraph.

Enterprise modeling started to be developed at the beginning of 80's, following the enthusiasm on the Computer Integrated Manufacturing (CIM) approach of the previous decade. In fact, when they have tried to apply the CIM concept to process and manufacturing industry, it became evident that a standard model of an enterprise was necessary to avoid the development of always-case customized software. One of the first proposals, coming from the Purdue University in the USA, was the Purdue Enterprise Reference Architecture (PERA). PERA was proposed by prof. T.J. Williams in 1990, under the Industry-Purdue University Consortium for Computer Integrated Manufacturing. Prof. Williams as a chemical engineer focused the PERA model on the process industry, also because at the beginning the CIM applications were mainly in the process industry, where the production volumes are massive while the system complexity is much lower than in the discrete manufacturing sector. The PERA reference architecture allows the modelling of the enterprise in multiple layers and in multiple stages of the architectural life cycle. Its three main building blocks are the Purdue Enterprise Reference Architecture, the Purdue Reference Model, and the Purdue implementation procedure. The equipment organization in PERA is made of: (i) Enterprise; (ii) Site; (iii) Area, which can be specialized in 3 different types: (a) Process Cell (divided in Units) for batch processing; (b) Production Unit for continuous operation; (c) Production Line (divided in Work cells) for repetitive and discrete operations. The development of PERA has continued and it is still active today thank to the supporting action of a consulting organization (www.pera.net).

Also at beginning of the 90's, under the push of Peter Bernus, Francois Vernadat and Guy Doumeingts, an IFAC – IFIP task force was created on Architectures for Enterprise Integration. The task force tried to create a first generalized enterprise architecture based on the previous efforts, also considering the discrete manufacturing environment. From this effort, the GERAM (Generalised Enterprise Reference Architecture and Methodology) originated. However, GERAM was not able to reach a remarkable diffusion and its development was subsequently abandoned.

These modeling approaches were of very high level, positioning themselves only at the upper levels of the enterprise information system. Thus, they were quite inappropriate for covering the automation level, where a more detailed description of the equipment and related control requirements are needed. In the meanwhile, the development of automation control systems and architectures made a stepwise improve-

ment thank to the development of the minicomputer technology (e.g. DEC PDP8 and 11 series computers). This trend invested both the process and the discrete manufacturing industry thank to the lowering of computer equipment cost, together with the increase in their power and scalability. Especially automotive manufacturers and white goods manufacturers invested a lot in the computer automation of their processes, applying such type of technology, during the 80ties and 90ties. Again, modeling and standardization emerged as crucial issues in this trend to enterprise automation. Main actions in these line where played by various organizations of different types. For example APICS (American Production and Inventory Control Society), a no-profit organization mainly focused on education and professional development, has played an important role as a reference body in the production management area (APICS, 1995). In 1992 MESA (Manufacturing Enterprise Solutions Association), another professional society, introduced the MESA model, last updated in 2006. The MESA model is based on three main components: the MESA Functions, the Context Model and the Collaborative Model. Focus of the MESA model is the integration of production information into the control and supervisory activity and the relationship with the various enterprise applications at the upper control level (i.e. Resource Planning, Customer Relationship Management, Product Lifecycle Management and Supply Chain Management) (MESA, 2003; MESA, 2004).

For what concerns standardization, the CEN (Centre Europeen de Norms) started issuing new standards related to the discrete manufacturing area through the activities of its Technical Committee TC 310, by publishing the CEN standards on Enterprise integration - Framework for enterprise modelling (ISO 19439-19440:2006-2007). In particular relevant standards in this set are (EN ISO, 2006; EN ISO, 2007):

- EN ISO 19439:2006 Enterprise integration - Framework for enterprise modelling,
- EN ISO 19439:2006/AC:2007 Enterprise integration - Framework for enterprise modelling,
- EN ISO 19440:2007 Enterprise integration - Constructs for enterprise modelling.

Instead, in the process industry the ANSI/ISA-95 or ISA-95, as it is more commonly referred (ISA, 2010; Brandl, 2008), emerged as an international standard for developing an automated interface between enterprise and control systems. This standard has been developed taking into account the previous work made on PERA and it can be considered as an extension of the PERA itself, for taking into account a global manufacturing perspective. ISA-95 still addresses the PERA physical model, which is based on batch, continuous and repetitive processes (ISA, 2010).

As a conclusion it is important to underline that enterprise modelling, at all various levels of the control architecture (i.e. corporate level, departmental level and operational level), is fundamental for establishing flexible and portable control applications. To this regard, the modeling of discrete production systems at the shop floor level is far away from reaching a well-defined and complete standardization. This is the area in which MSO (Manufacturing Systems Ontology, presented in detail in another chapter of this book) tries to give its contribution to development, by leveraging on previous work on the use of the object oriented approach in the description of manufacturing systems (Garetti et al., 1997).

7 Megatrends for future production systems

As it has been identified in the Factories of the Future 2020 document (EUR 24282, 2010), some *megatrends* have considerable impact and drive structural changes in nearly all manufacturing sectors. These megatrends can be identified as the following:

- Changing demographics (growing population, ageing societies, urbanization),
- Globalization & future markets (BRIC and beyond),
- Scarcity of resources (energy, water, other commodities),
- The climate change (increasing CO2, global warming, ecosystem at risk),
- Dynamic technology & innovation (ICT and virtualization, technology diffusion, the age of life science, ubiquitous connectivity, sensing and digitalization),
- Global knowledge society (know-how base, gender gap, war for talent, multiplication of data and information),
- Mass customization (personalized customization),
- Sharing global responsibility (shift to global cooperation, growing power of NGOs, increasing philanthropy).

Another report, "Factories of the future PPP strategic multi-annual roadmap" published by EFFRA (EUR 24282, 2010; EFFRA, 2012), stated that, in order to face the challenge of global competition, the European manufacturing industry will increasingly be forced to concentrate on specific issues providing competitive advantage through long-term innovation at the factory level. A key factor to strengthen European leadership in product and process engineering and in the development of manufacturing systems, both discrete and continuous, is considered to be the ability to achieve cost efficiency (including factors such as material supply, transportation, and cost of manpower), high performance and enhanced robustness, in a context of increasing product variability and continuously changing production volumes.

The contribution of Information and Communication Technologies (ICT) to manufacturing is seen crucial for improving the efficiency, adaptability and sustainability of production systems and their integration within agile business models and processes in an increasingly globalized industry. Also, it is underlined that the integration aspects of newly developed ICT into the production lines and the industrial environments requires complementary research and innovation efforts, concluding that these integration aspects will play a key role for generating and using smart production systems for factories in different industrial sectors.

8 Competitiveness through advanced automation[1]

Focusing on ICT and advanced automation as key enablers for European manufacturing competitiveness, hereafter, both the discrete manufacturing and the process sectors will be considered in sequence, taking into account the following perspectives:

[1] Part of the content of sections 8 to 14 is taken from the results of the Deliverable 2.1 of the Artemis EU JU eScop project.

1. **Market conditions** affecting advanced production systems (they can be considered as independent factors affecting the operating conditions of production systems),
2. **Production performance objectives** for the automation of advanced production systems (objectives that the control of advanced production systems should satisfy to cope with the market challenges), and
3. **Requirements** for the automation of advanced production systems (prerequisites that allow to the control solutions of production systems to pursue the above-mentioned objectives).

9 Market conditions for the discrete manufacturing sector

Given the megatrends presented in the section above and going in more detail considering companies based on discrete manufacturing (metalworking, automotive, white goods and so on), they are facing the following main challenges in their production activity:

- Variable demand: today, companies more and more need to be agile for successfully operating in a competitive environment in which market opportunities emerge and change continually with uncertainty,
- Frequent changes in operating conditions (cost of energy, material and labor),
- Competitive pressure from developing countries: firms in developing countries are under pressure to improve their performance and increase their competitiveness. New, low cost producers are entering global markets, intensifying competition in markets for labor-intensive manufacturers,
- Reduction of time to market of manufactured products: one of today's competitive environment's characteristics is the high speed with which products are designed, manufactured, and distributed which is forcing companies to search for innovative ways to do business,
- Opportunity to build up systems made of manufacturing equipment from different vendors: As the world becomes flat and boundaries break down, manufacturers need to understand the way to take advantage acting in such a diversified environment in order to obtain competitive advantage by keeping a systemic perspective,
- High demand in customized systems requires complicated and time-consuming re-engineering of products. The need for capable modular systems that enable fast re-configurations and interactions at small and big scale become more important.

10 Production performance objectives of the discrete manufacturing sector

The main performance objectives that the control of advanced production systems for discrete manufacturing should comply with to cope with the above-mentioned market challenges belong to the following categories:

- **Equipment level performance**

– Flexibility of the production equipment

Generally, it is possible to define the flexibility of a system under two categories: machine and routing flexibility. The inner category (machine flexibility) deals with the system's ability to change for producing new products and the ability of changing the order of operations executed on a single part. The latter category (routing flexibility) consists on the ability to use multiple machines to perform the same operation on a part, as well as the system's ability to absorb large-scale changes, such as in volume, capacity, or capability.

– Short ramp-up times

Changes in the level of market demand requires to a manufacturing enterprise to adequate its production level. Therefore, an enterprise with a short ramp-up time is able to change its asset configuration, or its organization, in a very short time, to properly follow and answer the changes in the market.

– Smart equipment

The concept of "smart equipment" defines an industrial machine equipped with industrial computer and/or embedded device, which is able to provide information - such as its status, location, working condition etc. - in order to reach asset awareness.

– Composability

A decomposable system provides recombinant components; therefore, it can be assembled in various combinations to answer to customer demand.

- **Plant level performance**

– Fast and cheap commissioning of new manufacturing systems

Commissioning of a manufacturing system can be classified into different activities such as design, procurement, assembly and installation, testing, operation and maintenance. A commissioning process can be applied to a new plant, but also to an existing one subject to expansion, renovation or revamping. It is not possible to summarize and describe the commissioning process in a few sentences; anyway, the enterprises are facing the challenge to reduce the time and costs related to commissioning.

– Plug and produce capability

The concept of "plug and produce" has been introduced in the production world after the consolidated development of the "plug and play" in the informatics consumer world. The enterprises feel the need to use the machines and production equipment in an easier way than today's condition, reducing the effort to configure, program and connect a new machine to an existing plant. The "plug and produce" concept introduces several requirements, e.g.: standardization in communication, open network.

– Easy re-configurability of the manufacturing systems

Manufacturing systems should allow *re-configuration* in order to facilitate changes in the plant and provide new and different services; the adoption of *composable systems* and *services* will allow introducing changes in the plant avoiding the re-programming of the involved devices.

– Scalability

It is important for a manufacturing plant to easily add and remove machines and software to increase or reduce the production capability in order to follow the real market demand, thus introducing savings.

– Overall Equipment Effectiveness

A better utilization of the asset (people, equipment and materials) can lead to an increase of the overall equipment effectiveness; this is enabled by low-level high-granular information elaborated in order to produce useful high-level indicators.

- **Control level performance**

– Interoperability at control, command and monitoring level

Here, the aforementioned aspects of modeling and standardization of enterprise architecture play a fundamental role for allowing the easy interfacing of control applications at the different levels of the management pyramid, thus going beyond the current situation of dispersed solutions and closed proprietary standards.

– Asset awareness

The business and management levels of an enterprise should be able to take decisions based on the most exact and updated information from the shop floor; this information can contain working load conditions, stock levels, health status, warnings and alarms, etc. The term asset awareness defines the capability of a system to provide the right information, at the right time, with the desired resolution.

– Interoperability

Machines, software and services from different vendors should be able to communicate and interoperate between themselves without difficulties thanks to the composition of heterogeneous network technologies and programming languages.

– Maintenance optimization-diagnosability

The maintenance optimization, and in a more spread view, the capability to diagnose the health status of a machine is enabled through the information obtained from the machine. The information, collected at a device level, allows implementing high-level services to optimize the management of assets.

– Customized manufacturing with late order freeze

Today's market condition requires short freeze period to enterprises; short freeze period is the period during which the customer cannot change a product parameter. The start point of the freeze period can be stretched to the point, where the parameter is actually incorporated in the product (order penetration point).

– Complexity management

The huge amount of data in an enterprise such as equipment condition, production status, stock levels, warnings and so on, introduces an unexpected degree of complexity, which has to be managed by proper tools.

– Reactivity

An unpredictable manufacturing market requires to take decision rapidly and then to act following the demand. An enterprise is able to survive and to win on this competition by reaching an enhanced internal reactivity.

- **Others**

– Reusability

According to cost reduction principle and sustainability enhancement a resource, such as a software, a device, or even a machine, should be reused.

– Intellectual protection

Enterprises pay lot of carefulness and attention in sharing information on their technologies and knowledge; therefore, it is important to take care of the importance of data protection.

– Cost efficiency

To obtain cost efficiency an enterprise should know and manage in an optimal way material costs (e.g. buying, stocking and maintenance cost), time-related costs (e.g. time spent during installation, movement or elimination of resources) and labor costs (e.g. personnel cost for installing, maintaining and producing).

– Business activity monitoring

The management of complex production systems and the today's market conditions requires having the right information at the right time, in the most restrictive case this means real-time. The huge amount of low-level information should be elaborated with almost no latency to produce meaningful and understandable key performance indicators for high-level enterprise function.

– Cross border integration

The integration between different level functions inside an organization (i.e.: engineering, production, management, etc.) is fundamental to increase enterprise performances. Nevertheless, it is important also to consider that "cross-border" integration is fundamental to obtain collaborative manufacturing in order to create competitive collaborations in highly competing environments.

– System usability

Manufacturing systems have historically not taken into account the needs of the user, only the needs of the system. While the need of information has always existed, little or no effort has been made to make users lives easier. By taking into account usability issues, performance in operator/machine manufacturing and problem solving can be highly improved.

11 Requirements for automation in the discrete manufacturing sector

Taking into account the above-mentioned objectives, the following requirements for the automation of advanced discrete manufacturing systems can be outlined:

- Decentralized automated control: automation systems become more and more complex, and their traditional hierarchical and centralized control may not be able to deal with this ever-increasing complexity satisfactorily. Decentralized approaches are considered promising, but they are not yet sufficiently understood for widespread industrial application (Kaindl et al., 2012).
- Use of the service oriented architecture (SOA) as a way to fully distribute and decentralize the control system architecture.
- Semantic description of the production system through the use of the ontology approach. The semantic description should be able to drive the configuration of the control system thanks to the interfacing with the service-oriented architecture of the control system kernel.
- Automatic update of system visualization after system reconfiguration: adding machines to a production plant generates the needs to provide information to an operator or a supervisor, so it is important to take in account not only the time for physical installation of the machine, but also the time needed for the configuration of the human-machine-interface.
- Knowledge driven configuration of control system: although web-enabled technologies are strengthening distributed automation in manufacturing, the pivotal technology will require a form of technical intelligence that goes beyond simple data, through information to knowledge. This will be embedded in manufacturing system components and within the products themselves, and will make it possible to meet agility in manufacturing over flexibility and reactivity, as addressed by the shifting Intelligence In Manufacturing (IIM) paradigm.
- Openness to third party services: in order to gain advantages, such as efficiency, savings and high quality production, enterprises need highly qualified capabilities to elaborate data and take decisions. These capabilities can be often found outside of the enterprise itself, and in particular for SMEs, it is not possible to acquire the resources needed to provide highly qualified services. The possibility to find and buy services from outside will help in gaining advantages without high costs; on the other side an enterprise can also increase its market by offering new services.
- Standard and open communication infrastructure: much work has recently been done to establish open standards promoting technology connectivity and machine-to-machine communications. Developments in open architecture standards and communication protocols will serve to facilitate automation through promotion of "plug and play" technologies, which can be offered from a wide variety of sources. The MTConnect Institute has developed open (non-proprietary) and royalty free communication standards based upon the Extensible Markup Language (XML). These standards allow for machine-to-machine communications and promote interoperability between existing technologies. Similarly, the OPC Foundation offers seven open communication specifications that also promote connectivity and interoperability. Research regarding machine-to-machine communication is also common in the academic realm. Cyber OPC is a dedicated protocol developed for communication with CNC machines over public networks. The use of STEP-NC is discussed as a communication language between the shop floor and the plant scheduling level. Additionally, Web Services (machine-to-machine communication

over the World Wide Web) for the development of distributed manufacturing management frameworks is a core issue of today research (Wu et al., 2013).
- User and context awareness: recent efforts have been made towards including the user state and his or her context into the operational behavior of manufacturing systems. While great, and needed effort has been placed into machine-to-machine communication little effort has been placed on machine-to-human interaction. By means of knowledge driven technologies, as well as recent improvements in the sensing of the physical and emotional state of humans it is possible to reach a new level of synergy between users and machine by making the machines be aware of the human. While before the human only responded to the machine's state and behavior the relationship can now be pushed both ways.

12 Market conditions affecting the process industry[2]

The main market factors affecting process industries in recent years are as follows:

- **Product quality and plant availability**: in the highly industrialized countries, process automation serves to enhance product quality, master the whole range of products, improve process safety and plant availability.
- **Efficient usage of resources and lower emissions:** current regulations are stricter than before (e.g. wastewater management, safety, etc.).
- **Mass production**: in the rapidly developing countries, mass production is the main motivation for applying process automation.
- **Higher level of automation is required**: many process industries are moving to countries with lower labor cost. In Europe, it is becoming highly important to not depend on labor and focus on highly capable automation.
- **High quality standards and no room for scrap**: the current process industry standards are rigid. Whatever is produced has to be high quality with the minimal amount of scrap production possible (e.g. impurities in steel, low quality paper).
- **Remote market**: currently, many different types of process industries operate in Europe, but most of the market is abroad. Manufacturers face the question of moving their production abroad, exporting raw materials or exporting final products.
- **High power consumption:** the process industry consumes much more energy than the discrete manufacturing industry, this makes it highly dependent on the cost of power and taxes (e.g. UPM Jämsä uses more than 200MW when running, i.e. approximately 50% of the power produced by a nuclear reactor). The greatest demand for process automation is in the chemical industry, power generating industry, and petrochemical industry; the fastest growing demand for hardware, standard software and services of process automation is in the pharmaceutical industry. The importance of automation continues to increase in the process industries.

[2] Thanks are due to Roberto Camp from FluidHouse for his contribution to the sections regarding the process industry

13 Production performance objectives of the process industry

The main performance objectives that the control of production systems for the process industry should comply with, to cope with the above-mentioned market challenges, can be grouped into the following categories:

- **Equipment level performance**

– Short ramp-up times
  In the same manner as the manufacturing industry the changes in the level of market-demand requires to a manufacturing enterprise to adequate its production level. An enterprise with a short ramp-up time is able to change its asset configuration, or its organization, in a very short time, to properly follow and answer the changes in the market. It is important to note, however that the time scales between both industries are quite different.

– Smart equipment
  Same as for the discrete manufacturing sector.

- **Plant level performance**

– Scalability
  Same as for the discrete manufacturing sector.

– Overall Equipment Effectiveness
  Same as for the discrete manufacturing sector.

– Product feature flexibility
  Process industry plants must be capable of using the same systems to produce products with different quality and different features depending on what product needs to be made. Production change-up times must be as short as possible.

- **Control level performance**

– Interoperability at control, command and monitoring level
  Same as for the discrete manufacturing sector.

– Asset awareness
  Same as for the discrete manufacturing sector.

– Interoperability
  Same as for the discrete manufacturing sector.

– Maintenance optimization-diagnosability
  Same as for the discrete manufacturing sector.

– Complexity management
  Same as for the discrete manufacturing sector.

– Reactivity

An unpredictable manufacturing market requires to take decision rapidly and then to act following the demand. An enterprise is able to survive and win on this competition by reaching an enhanced reactivity of the overall enterprise. This is a particularly difficult challenge in the process industry (e.g., one cannot change a machine that produces newspaper, to one that produces toilet paper without great cost and time).

- **Others**

– Intellectual protection
  Same as for the discrete manufacturing sector.

– Cost efficiency
  Same as for the discrete manufacturing sector.

– Business predictability
  Given the slower change and higher cost that the process industry incurs it is necessary to have a good future view of market evolution (at least 5 years). This enables companies to prepare for upcoming market demands.

– Cross border integration
  Same as for the discrete manufacturing sector.

– Updateability
  The lifetime of process industry facilities is much longer than those of the discrete industries. As such being able to update 20-30 year old systems is a demanding task that requires the integration of modern modeling and simulation tools. There is not room for mistakes as any change can have great cost (e.g. moving a paper machine to another country, or updating it, or updating only part of the current machine).

14 Requirements for the automation of process production systems

The general requirements for automation of process production systems are quite similar to those requirements that have been described previously for discrete production systems. This is because the features of flexibility, distributed control, semantic rich description of reality, awareness and advanced human machine interfaces, allowed by new ICT technologies, which are so important for the discrete production area, can also give substantial benefits also in the sector of process production.

15 The new paradigm of Open Automation Manufacturing

In factory automation, the evolutionary path of real-time control architecture is directed to overcome the rigidity of current traditional solutions at shop floor level. In fact, the current architecture of MES real-time control is based on fragmented and scattered information from the field, joined with a rigid hardware structure, hence being suitable and efficient only in stable contexts.

A new paradigm called Open Automation Manufacturing has been proposed within the eScop research project for defining technologies that can be helpful for building automation systems that are flexible, reconfigurable, smart, self-aware, capable of plug-in and add-on. This paradigm aims at overcoming the current drawbacks for the shop floor control level of automated production systems, thus improving the state of the art of the overall architecture of the manufacturing production control. This goal is pursued as a tentative answer to the requirements for automated production that have been discussed previously for allowing the European industry to stay competitive on the global market.

To this regard, the traditional architecture of production control for discrete manufacturing will be compared in the following with the proposed architecture of open automation manufacturing.

- **Traditional production control in discrete manufacturing**

The current architectural solutions for production control in the discrete manufacturing sector are schematically represented in Fig. 2. The manufacturing execution system usually requires various hardware and software hierarchical levels, ranging from low-level controllers for the physical devices to higher-level controllers for central coordination.

The production sequence, that drives local controllers, is generated dynamically at run time by the MES supervisor, which assumes the role of a synchronizer. The system control logic can assume different structures: it may vary from highly centralized solutions in which the supervisor hierarchically controls simple control units, to more distributed solutions in which, for example, the CNCs of the machines are enriched with functionalities dedicated to the control of the production system.

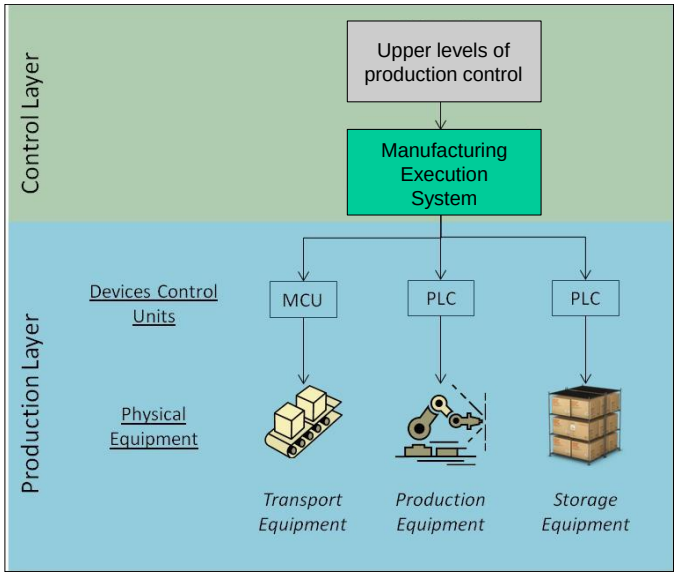


Fig. 2. Current architecture for production control for discrete manufacturing

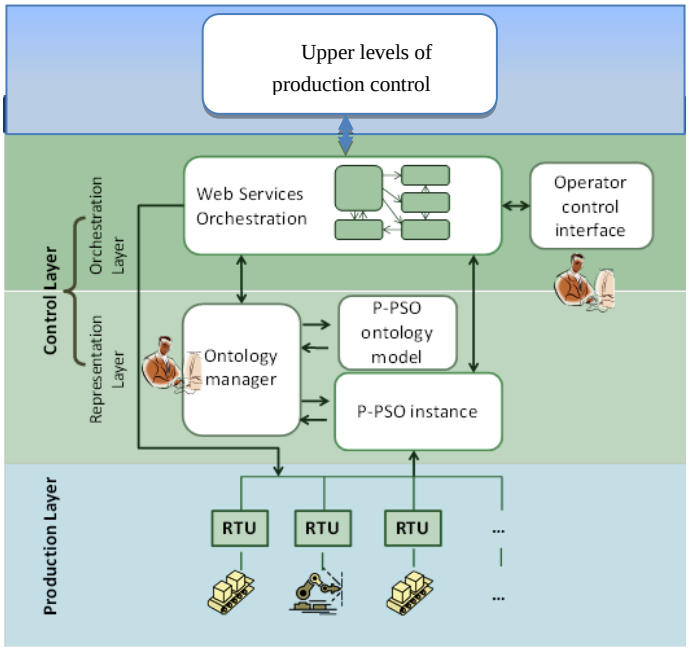


Fig. 3. Concept of the new control architecture of the eScop project

- **New architecture for production control in discrete manufacturing**

The central concept of the new architecture proposed by the eScop project is to combine the power of embedded systems with an ontology-driven service-based architecture for realizing a fully open automated manufacturing environment (Fig. 3). The true innovation of the proposed solution is the merge of the power of ontology knowledge and SOA control approaches that allow the control system to be automatically configured by the ontology knowledge content, while embedded systems and SOA allow the architecture to work. The result is a modular, fully open solution for the operational control of manufacturing equipment allowing:

- easy and fast commissioning of new plants,
- achievement of "plug & produce" inclusion of new equipment,
- replacement of the traditional control solution, based on hierarchical hardware architecture, by a single level cohort of embedded systems and a series of software control levels.

References

APICS (1995). APICS Dictionary. APICS, Falls Church, 1995.

Brandl, D. (2008). What is ISA-95. Industrial best practices of manufacturing information technologies with ISA-95 models. IEC65E/JWG5 Convener ISA 95 Editor, BR&L Consulting.

EN ISO (2006). EN ISO 19439: CIM System Architecture-Framework for Enterprise Modelling.

EN ISO (2007). EN ISO 19440: Enterprise integration - Constructs for enterprise modelling.

EUR 24282 (2010). Factories of the Future, PPP Strategic Multi-annual Roadmap, Publications Office of the European Union, Luxembourg.

EFFRA (2012). Factories of the Future 2020, Factories of the Future Public-Private Partnership Roadmap, EFFRA.

Garetti, M., Bartolotta, A., Corradi, E., Rabe, M., and Raimondo, A. (1997). Design issues of an integrated software workbench supporting the manufacturing systems design process. *Proceedings of the International Conference Computer Applications in Production and Engineering (CAPE'97)*: 320–329.

ISA (2010). ANSI/ISA-95.00.01-2010, Enterprise-Control System Integration, International Society of Automation.

Kaindl, H., Vallée, M., & Arnautovic, E. (2012). Self-Representation for Self-Configuration and Monitoring in Agent-Based Flexible Automation Systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 43(1): 164-175.

MESA (2003). Collaborative manufacturing explained. http:// www.mesa.org/.2003. Accessed 2014.XI.12.

MESA (2004). Mesa's next generation collaborative MES model. MESA White Paper. http://www.mesa.org/.2004. Accessed 2014.XI.12.

Wu, D., Greer, M. J., Rosen, D. W., & Schaefer, D. (2013). Cloud manufacturing: Strategic vision and state-of-the-art. *Journal of Manufacturing Systems*, Vol. 32(4): 564-579.

Ontology-Aided Manufacturing and Logistics

Stanisław Strzelczak

Warsaw University of Technology, Warsaw, Poland

s.strzelczak@wip.pw.edu.pl

Abstract. This chapter reviews advantages of applying the means of ontology engineering for manufacturing and logistics operations management. The eScop approach is considered as a reference framework for the discussion. Semantic integration is recognized as crucial for those domains, which intensively exploit information and communication technologies (ICT) and automation technologies (AT). The distinctive provision by ontologies is the ability to reason. Ontology engineering can facilitate new forms of organizational structures and processes. Furthermore, intra- and inter-organizational integration of different layers, functions, (sub-)systems, processes and domains may be simplified or enabled. It is advocated that the significant advantages of the eScop approach go far beyond its original focus, which was on the re-configurability of automated manufacturing systems. Extensions and adoptions of the eScop approach for manufacturing and logistics area are investigated and conceptualized herein.

1 Introduction

The main streams of industrial development in recent decades were facilitated to major extent by innovations in the ICT and AT area (Fig. 1). The key enabling technologies with this regard were: Web and Cloud services, smart and mobile devices, Big Data technologies, Service-Oriented Architectures (SOA) and semantic technologies. In parallel, mostly due to globalization and technology revolution, increasing openness and networking were affecting various areas of civilization, hence enhancing different aspects of quickly growing complexity and variability, like: spatial spread, interdependencies and couplings, uncertainty and turbulences, etc. The above named qualitative changes tended to act as generic determinants for the developments considered herein, including ontology-aided manufacturing and logistics. This applies in particular to those systems and processes that integrate human and technical components, being used in various areas, like: communication, social life, commerce, finance, industry, healthcare, legal system, urban life, weapons et al. Therefore extraordinary challenges and opportunities arise concerning development of systems, resources and processes. Shortening life-cycles, increasingly frequent change and growing overall variability were giving rise to importance of structural and process changeability, hence defining crucial operational features required from systems and processes. Handling various aspects of complexity and changeability provides a particular issue for management of globalized operations in manufacturing and logistics.

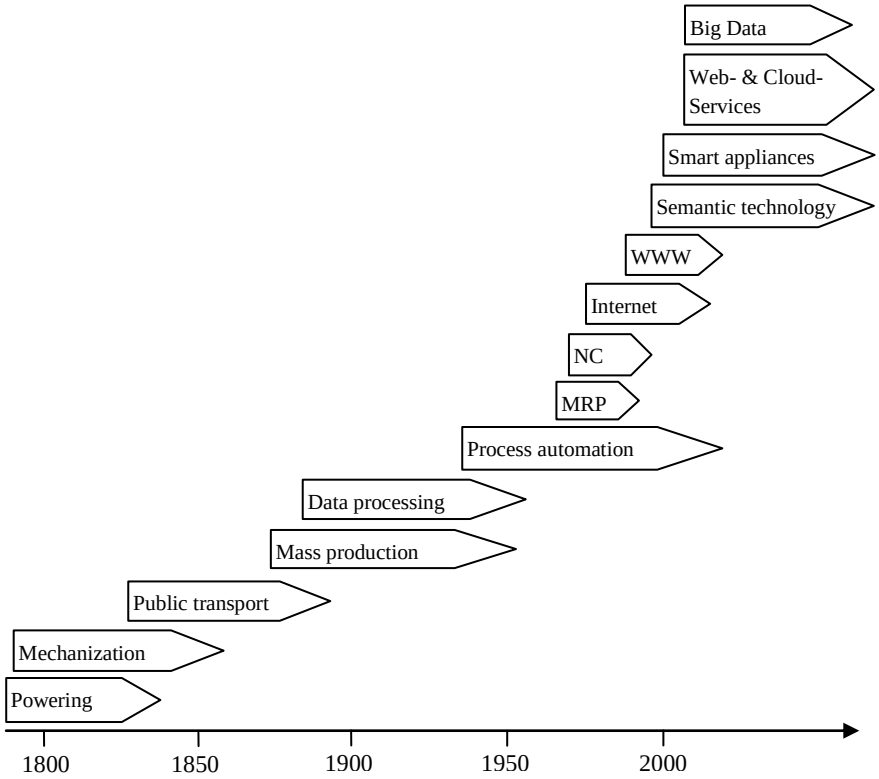


Fig. 1. Mainstream technologies and developments supporting industrialization

This chapter investigates opportunities from using ontologies in manufacturing and logistics. A twofold approach is applied with this regard. Firstly, to recognize the needs and requirements as realized by industries. Secondly, to analyze enabling potential and key development challenges in reference to the abstractions and means of ontology engineering, when addressing production resources, processes and systems.

The first perspective applied is mostly derived from reflecting on outcomes from two research projects. One of them is a part of the ongoing EU eScop project[1], which focuses on improved re-configurability of automated manufacturing systems by means of ontologies, SOA and embedded devices. The second research was a part of another project, which centered on new ideas for industrial engineering, particularly in reference to future Webs-driven production environments that could be operated by combination of distributed intelligence and smart appliances and managed in novel ways using non-hierarchical structures for planning and control of operations[2]. Both projects used the insights from needs and requirements as understood by industries.

[1] EU JU ARTEMIS Program, Project no. 332946 "ESCOP - Embedded systems for Service-based Control of Open manufacturing and Process automation".

[2] PSP 504/01518/1103/40.000105 grant of Warsaw University of Technology "Management and assessment of novel technical & organizational solutions".

The second perspective applied is primarily based on conceptual research. It was also supported by literature review of those existing concepts that consider ontology-aided management and operation of manufacturing and logistical activities. The concern herein is on creative ideations of future advantageous operational environments for the domain. It is assumed that they could rely on available technologies, or at least on those of them that are about of reaching maturity level. Alternatively they could be built upon novel ideas concerning systems or infrastructures, which would eventually request for some realistic technological innovations. How much an aim of that kind may be challenging, can be learned from the story of Facebook. It has quickly revolutionized global social communication, but simultaneously it has provided new type of infrastructure for novel business processes, e.g. active marketing. Similarly, ongoing developments concerning e-commerce, Web browsers, smart mobile appliances and tracking technologies tend to revolutionize retailing, hence eventually systems and processes of supply chain management and logistics.

Using both insights derived from two different streams of research a range of ideations for ontology-aided manufacturing and logistics was identified and conceptualized. All of them are rooted in the eScop approach, which is presented as much extendable and of much more general use, than it was originally presumed. The ideations exploit this important advantage and go far beyond the original eScop framework. Therefore they indirectly set directions for eventual further research, including practice oriented developments. All the conceptualized ideations are presented in section 5 of this chapter.

2 Development needs and requirements as viewed by industries

This section reviews the needs and requirements concerning development of open knowledge-driven manufacturing and logistics, which could be supported by ontologies and (Web-)services, as they are realized and understood by industries. The basis for this purpose are results of performed research, which had two key objectives:

1. to identify future needs and requirements of European industries concerning operations management and smart automation in the area of manufacturing and logistics,
2. to identify existing gaps within current solutions for accomplishing European manufacturing requirements for the future.

The research has also aimed at qualitative forecasting of technologies by incorporation of a foresight element. Hence a creative capacity of interviewed experts has been expected and stimulated along the research.

From the available methods of research semi-structured interviewing of experts from various industries was chosen. The reasons for that have been as follows:

- to protect high dependability of results;
- to use the expertise of interviewed experts in an interactive mode;
- to drive creative thinking of experts while assessing and forecasting;
- to get high value-effort ratio along the research.

Each company incorporated into the research was approached by phone. Then a formal letter was e-mailed, with a brief description of the eScop project and a list of guiding questions as attachments. Finally meetings were arranged at the company site. Later the results of each interviewing were initially processed, than compiled and processed again. During the meetings questionnaires provided in advance have been used to lead the interviews. Using the questions from the questionnaire together with other direct, indirect, and ad hoc questions, the expertise could be extracted and many assessments, forecasts and some advices and suggestions could be obtained. Some kind of brainstorming was normally happening in case of a need for creative thinking. All interviews were precisely documented, and in most cases audio-recorded.

Considering the purpose of the research the following categories of companies were identified as those having expertise in the area of interest:

- companies that use some kind of ICT and automation of manufacturing and logistical processes;
- manufacturers of automated equipment for manufacturing and logistics;
- integrators of manufacturing systems and industrial supply chains;
- providers of ICT solutions for automation of manufacturing and logistics.

The expertise from the telecom sector concerning architectural and communication solutions was also considered as important for the research, as it is directly involved in development of technologies used for Web services.

The research targeted mostly manufacturers, of those both, discrete and process industries. Among discrete industries a particular attention was given towards automotive sector which is being considered as a cutting edge in terms of system solutions and management practices. The following sectors of process industries were approached: petrochemical, chemical, cement, electro-power generation. Geographically, organizations and plants located in the following countries were researched: Czech Republic (2), Finland (2), Italy (2), Germany (4), Poland (8), Peoples Republic of China (1), Spain (1), Switzerland (1). By ownership type half of the companies were national and half multi-national. By size, five companies can be classified as SMEs, and 16 as non-SMEs (of them vast majority belonging to the globally operating big groups, mostly blue chips).

The data about researched companies, like their type and sector, is presented below in Table 1.

Considering the goals of the discussed research following categories of professionals were approached as experts:

- in case of manufacturing equipment providers, those professionals in charge of (or involved in) products development and competent in the automation / CNC aspects of the products;
- in case of users, people in charge of facilities selection and design, like process engineers, and also maintenance specialists;
- in case of integrators and ICT solutions providers, people in charge of products development, sales and customer services.

Table 1. Companies researched by sector, category and type

No.	Name	Sector	Category	Type
1	TRW	Automotive	User	Discrete
2	TrelleborgVibracoustic	Automotive	User	Discrete
3	Fly Polska	Aviation	User	Discrete
4	Fluidhouse Oy	Machine building	User	Discrete
5	Monosuisse	Chemical	User	Process
6	PKN Orlen	Petrochemical	User	Process
7	CRH	Cement	User	Process
8	Elektrownia Bełchatów	Power generation	User	Process
9	KraussMaffeiTechnologies	Machine building	Provider	Discrete
10	Netstal	Machine building	Provider	Discrete
11	Berstorff	Machine building	Provider	Discrete
12	KraussMaffeiTechnologies-RPM	Machine building	Provider	Discrete
13	KraussMaffei Automation	Machine building	Provider	Discrete
14	SCM	Machine building	Provider	Discrete
15	INCAS	NA	Integrator	Services
16	ICONICS	NA	ICT developer	Services
17	REX Control	NA	ICT developer	Services
18	THT Control Oy	NA	ICT developer	Services
19	Ericsson Spain	Telecom	ICT developer	Services
20	A & J Solutions	Telecom / ICT	ICT developer	Services
21	National Key Laboratory on Remanufacturing	R&D	R&D	Services

The structure of interviewed experts in terms of rank and respective type of expertise is presented below in Table 2.

The lists of questions provided in advance to the researched companies were tailored according to the company category. The questions were focused around core

Table 2. Rank-based structure of interviewed experts

No.	Rank of expert	Number	%
1	CEOs	4	5
2	CTOs	5	7
3	Other CxOs	1	1
4	Σ CxOs	10	13
5	Middle level managers	15	20
6	Engineers	47	67
	Σ	72	-

novelties expected in terms of open knowledge aided operations management, particularly including the following aspects:

- New systemic solutions in terms of structures, layouts, processes, flows, planning & control principles and architectures, etc.;
- New functionalities, internally and externally provided;
- Novel hardware and software architectures;
- HMIs;
- Handling complexities; possible or required limitations and couplings;
- Possible insights for development of ontologies;
- Knowledge management and cognitive aspects;
- Interfaces with other domains;
- Business models and IP related issues;
- Maintenance of knowledge / ontologies.

The spread and variability of opinions was handled by a subjective assessment and synthesis. Some controversial opinions were not ignored but cited. Although the number of interviewees was moderate, using statistical characteristics as a basis for conclusions was assessed as unjustified.

Most interviewed experts named language(s) of communication, and particularly semantics, as the key to successful open knowledge-driven manufacturing and logistics. Some interviewed directly suggested ontologies as the generic novelty, that could be used to describe all information, knowledge and models exploited or used along operations management. Therefore they should support exchange of all types of data, information and knowledge, this way enabling semantic integration of applications and system components. In particular, ontologies are also expected to facilitate new functionalities built upon semantic processing capacities.

Ontologies could also support new ways of managing and controlling operational processes and also development of them in a dynamic run-time mode. Embedded appliances equipped with standard interfaces could be built into all items subject of management and control within manufacturing and logistical systems and integrate them into one open knowledge driven system.

Concerning the interoperability of items within easily connective systems and other environments it was suggested, that apart of material or “real” elements (facilities, transported items, documents), virtual temporary items should be eventually considered (i.e. perdurants in meta-ontological terms). These could be occasionally generated and exploited along operations performance or management. Examples could be: virtual subsystems and capacities, spatiotemporal demand structures (schedules, projects, orders, batches, tasks), virtual (sub-)processes and flows (possibly all “moving” and “trackable” items should be subject of such considerations, e.g. assembly kits), spatiotemporal interdependencies concerning coordination patterns (e.g.: synchronizations, time-rigidness etc.), components of tacit knowledge and all other required elements of knowledge, etc.

Concerning functionalities that could be subject of knowledge-based automation, the following were named as the most basic and required:

- direct communication between devices, e.g. machine-to-machine;
- local and/or heterarchical control of operations;
- acceptance, issuing and initiation of orders;
- smart dispatching, based on flexible processes and routings;
- local response to various breakdowns or unplanned circumstances.

Among potential welcome "smart" abilities of embedded appliances the following were often suggested:

- aggregation of data to support locally various inferences; alternatively, communication with "clouds" to get from them similar services;
- local adjustments or adaptive controls according to current circumstances;
- local diagnostic and early-warning capacities.

Many companies suggested that future embedded appliances should perform some functionalities of Total Productive Maintenance as well as intelligent process control. E.g. early-warning against possible break downs of equipment or failures of process could be enabled. Big Data mining of process data could be realized in clouds, this way enabling better process planning or control, failures forecasting etc. Finally, process routings could be rough or flexible. They could be later extended and tailored by smart appliances to adapt to local characteristics of current workplace. Following the above, productivity management activities could be also supported. E.g. records from processes could be mined in clouds to provide well-justified time standards or to indicate potential for improvements.

Concerning control principles for flows of operations, processes and items, the following suggestions were given:

- each device equipped with limited own intelligence, at least to support self-adaptation to current local changes;
- ability of smart bulking, issuing, accepting, initiating, prioritizing and dispatching;
- learning capacity concerning "bad" and "good" processes and routings;
- ongoing consideration of performance goals along scheduling or dispatching;
- adaptive management of bottlenecks, blockings, starvations etc.

Local orchestration of operations should be based on online communication between system and devices. Alternatively external orchestration could be provided by cloud services. Furthermore, the following approach was also suggested by some interviewees. Embedded devices or software agents have own orchestration intelligence. They could communicate each other and decide solely, through bargaining, auctioning and similar mechanisms, in which way to produce and forward given orders. Finally, flexibility of processes and routings was stressed. This functionality was described by one of the interviewees by a comparison of train to a car – the car can change the road during a travel when needed. Flexible routings, priority based scheduling, auction based alignments could be applied for the above purposes. Scheduling could be considered as a part of supply chain. Centralized scheduling could be applied in particular cases. Normally it should be possible to override decisions by the schedule from one centralized system, except legacy systems or legal/safety restrictions.

Two other features of management and control of operations were also suggested by many interviewees:

- possible use of web based interfaces (appliances and users should be able to operate from any device, e.g.: laptop, tablet, phone etc.);
- the logic of systems and processes should be stored in knowledge bases, not in applications.

Concerning advantages of functionalities to be eventually obtained by exploitation of tracking technologies, following capacities were suggested as particularly important:

- intelligent reporting or notifications about achieving point of progress in process or along forwarding (clients / suppliers);
- possibility to writing route to the local memory of moving item (e.g. RFID);
- using creatively data about exact position; e.g. it could enable counting products on exact palette or provide possibility to read a real-time position of each palette in a production line; each of those data collected can be stored and then analyzed in case of various accidents disrupting production or transport.

Some of the interviewed companies were suggesting that a clear distinction of time-rigidness and time-criticality of different tasks is crucial to obtain relevant reference solutions. Hence ontologies should incorporate means to represent such a kind of requirements.

Interesting insights to the research have been provided by companies from the telecom sector. This sector is highly advanced in research which is aiming at integration of different technical equipment, not just mobile phones, by internally developed technologies. IP based integration of technical devices is of particular R&D interest of this sector. From the information provided by telecom experts it appears, that it is technically possible to provide a smart controlling device, which may be equipped with the following features:

- use of any interface and interfacing protocol;
- use of different modes of external communication; some of them are much more efficient than those commonly used by industrial automation solutions[3];
- use of many operating systems;
- store locally a lot of data;
- built-in intelligence;
- can communicate directly by Internet.

The telecom experts also recommend technologies developed by their sector, which can support controls under non-deterministic circumstances, e.g. failures or

[3] A collection of mobile phones can operate like an independent network, as all of them receive and send signals, possibly in a peer-to-peer mode, depending on the software built into them. This mode of communication can be considered as an alternative safe mode of data exchange, e.g. internally.

breakdowns. For these purposes the Erlang open-source language has been developed, which targets programming of non-deterministic controls, e.g. considering breakdowns. To underline, even now embedded systems can be provided with much stronger capacities comparing to average PLCs or RTUs, and much cheaper.

Many interviewed suggested like an open source solution to production and logistics ontologies, however noticing the issue of who and how should maintain them. Other experts suggested that it might be impossible to develop a unified domain ontology enabling to handle unique control models for operations management, particularly to handle variability and complexities between processes and elements of manufacturing and logistical systems, while core shared ontologies could be a realistic solution.

The above review of managers' opinions justifies the following summary:

1. Industries exhibit a good understanding of the potential of semantic technologies, Web services and other technologies that can lead towards open knowledge-driven manufacturing and logistics. The theoretical potential of these technologies is welcome and appreciated.
2. The provisions that are expected from the new technological developments fits very well to the needs and requirements of industries. The expectations are both, exact and well founded.
3. The gap between the available provisions and the needs and requirements defines a clear and sound objective for the future developments.

However, for what concerns implementations, the interviewed managers provided a consistent and valid pattern of recommendations, which is as follows:

1. As long as the eScop approach does not provide some trustworthy reference points, and cannot be recognized as a robust technology (i.e. according to the standards of approval, as commonly used by industries), its implementation must rely on partial solutions, i.e. overlaps to the existing solutions.
2. The partial solutions should be centered around two possible ways:
    a. The partial solution is an overlap that provides a particular functionality which is not offered by existing MES solution, and cannot be added due to some barriers.
    b. The partial solution is an overlap that provides an additional functionality that is of high importance for the user, but cannot be added to the existing MES solution due to technological barriers. For example, it could be an additional control, based and on new or existing sensors and actuators. If such additional control provides some important advantages, like e.g. energy or materials savings, or quality improvement, or reduced emissions, than the approval of eScop approach will be easy.
3. The exceptions to the above rule might be full implementations in SMEs. In some situations these companies may be far more open than the blue chips, to accept a complete eScop implementation, i.e. the MES system.

3 Literature insights on manufacturing and logistics ontologies

The number of published ontologies for manufacturing and logistics is limited. Many of them target specific areas, and only few address management and control manufacturing and logistical operations. All these are reviewed below. An earlier literature research was also used as the input herein (Grubic and Fan, 2010). Only the most representative papers from different research centers working on production and logistics ontologies are refereed below.

The Enterprise Ontology (EO) (Uschold et al., 1998) was developed aiming at: (i) enhance communication between humans; (ii) provide a basis for specifying applications; (iii) support interoperability. The EO consists of five parts: (i) meta-ontology and timing; (ii) activities, plans, capabilities and resources; (iii) organization; (iv) strategy; (v) Marketing. Although some concepts which form the EO are generic to many organizations, the remaining concepts address enterprise constructs (e.g. non-legal ownership) that are at the very high level.

The ontology of Soares et al. (2000) focuses on production planning and control in a virtual enterprise to improve human communication and to support specification of system requirements. It is founded on meta-ontology derived from the EO ontology, whereas the concepts are defined by natural language and object models. These were grouped under three main sections: (i) networked/extended organizations; (ii) plans; (iii) orders management.

Lin et al. (2004) developed the MSE (Manufacturing Systems Engineering) ontology to support intelligent coordination in extended or virtual enterprises. The MSE ontology conforms to a simple taxonomy of various concepts, like: strategy, project, enterprise, extended enterprise and resource. It exploits the MSE Moderator, which is 'an intelligent support application designed to facilitate and improve concurrent engineering design by enhancing the degree of awareness, cooperation and coordination among engineering team members' (Lina and Harding, 2007). The MSE ontology has seven top-level classes that are further detailed and classified within a hierarchy of subclasses. They are: (i) project; it represents flows of material and other items during operations of an extended/virtual enterprise; the latter are linked by process class that represents a transformation which in turn is enabled by different resources modelled by the resource class; (ii) flow; (iii) process; (iv) enterprise; it provides a structure for managing processes and resources by employing different items that belong to the strategy class; (v) extended enterprise; it represents the aggregation of different enterprise objects; (vi) resource; and (vii) strategy.

Madni et al. (2001) introduced the IDEON ontology to support design, reinvention, managing and controlling collaborative distributed enterprises. IDEON integrates multiple perspectives to capture different concepts and relationships which describe an enterprise. Each of these views is represented as a separate Object-Oriented model (using the Unified Modelling Language) and conforms to simple taxonomies of resources and activities. The four views are: (i) enterprise context view; it is used to represent the interaction of an enterprise with its environment; it consists of several concepts that deal with observing and assessing the state of an environment; (ii) enterprise organizational view; it focuses on the organizational structure of an enterprise

and deals with lower level concepts (e.g. goal, strategy, objective, process, or person) than the enterprise context view; (iii) process view; it is used to equip the ontology with concepts required to represent the (re)planning-execution-control cycle; the process concept is further classified into three sub-concepts: planning process, plan and activity; (iv) resource/product view; it details types of resources required for execution of processes.

Dassisti et al. (2008) proposed an ontology-based model, following the IEC 62264 standard (2002). It includes: (a) Product Definition Model; (b) Material Model; (c) Equipment Model; (d) Personnel Model; (e) Process Segment Model: it contains process segments that list the classes of personnel, equipment, and material needed; (f) Production Schedule Model: it shall be made up from one or more production requests; (g) Production Capability Model; (h) Production Performance Model. Extended conceptualizations were provided for two classes.

Lemaignan et al. (2006) presented MASON ontology (Manufacturing's Semantics Ontology), which is built upon three head concepts: resources, operations and entities. For each class several subclasses were defined. The ontology is based on the Web Ontology Language (OWL).

Cândido et al. (2007) described the ontology for shop floor assembly. Two categories of concepts were proposed: modules and skills. Modules represent physical processing units or their aggregation and they are compositions of workstations. Workstation is a composition of units (transforming, flow and verification unit). Two common constructs "composed-of" and "is-a" are used to describe compositions and specialization relations. Skills represent abilities to perform operations. The basic element in the system, which uses ontology as a data model for reasoning about objects and their relations, is the Manufacturing Resource Agent (MRA). This agent searches ontology after instantiation for skills it supports, using their serial number and equipment type. Then it registers its capabilities in the yellow pages provided by a special Directory Facilitator (DF) agent, which manages and provides information about services provided by agents. MRA agents can form coalitions to provide combined skills. In such case there is a Coalition Leader Agent, which registers in the DF all complex skills provided by coalition and coordinates execution of elementary actions by particular coalition members.

Obitko et al. (2010) proposed an ontology for Agent-Based Manufacturing Systems (2010). The basic categories within it are: (a) customer order, (b) production plan, (c) workstation, transportation and material handling. All of them reuse classes and properties from the proposed Core Ontology, which for example separates physical and information resources. There are also some other ontologies suggested, such as the ontology for the configuration of the system.

Battista and Giordano (2010) proposed a modeling framework which incorporates: (1) product and planning data structures (BOM (Bill of Materials), process charts, MPS (Master production Schedule), calendars, etc.); (2) data on operations and equipment; (3) production and inventory management control policies; (4) distinction between physical and information layer. Although the above framework is not directly named as ontology, the authors advocate for using ontology of manufacturing system including the proposed elements.

Garetti and Fumagalli (2012) suggested three layers in their P-PSO ontology, i.e.: physical, technological and control. The main classes in this ontology are: part; component (system structure); operation; controller (decisional element performing functions of production planning and control; it can be person, PLC or software); operator; subsystem (service class allowing grouping of objects of classes in a nested way). The primary sub-classes of the component class are: processor, transporter and storage. The secondary sub-classes are: tool, fixture and unit load (i.e. entity used to move or handle parts). The ontology includes a sound taxonomy of the transporter and storage sub-classes. Apart of the controller class the control part of P-PSO ontology uses the following classes: rule (logic of decision making for controller; it can be algorithm, heuristic, simple rule or knowledge-based rule); order (part and quantity to be produced or purchased); production plan (set of orders generated by controller for time frame; it can be divided into sub-plans); batch (it does not correspond to a unit load); task (specific action of controller on component, part and operation classes, i.e. the translation of controller actions at the physical level; examples are: routing, process selection, dispatching).

Krupa (1984, 2006)[4], developed in 1980s a complete conceptual framework for manufacturing and logistics rooted in the theories of sets, graphs, automata artificial intelligence formal linguistics. It uses two basic categories for describing the domain, i.e. resources and tasks. It is distinctive by many features, of which some were never before addressed by the literature. The key of them are as follows:

1. Semiotic interpretation of resources, namely, in terms of classes, objects and denotations.
2. Distinction of transformation operations on the resources.
3. Distinction of transformation-informative, structural and functional relations of resources.
4. Consideration of other structuring formalisms for resources than those rooted in the theory of sets, e.g. collectivities of resources.
5. Distinction of systemic transformation operations on the resources.
6. Functional and automata-based interpretation of dynamic behavior of resources.
7. Distinction of global and local (a priori and a posteriori) discrepancies between properties.
8. Distinction of tacit (procedural) and explicit (structural) representation of tasks.
9. Distinction of different forms of representation of tasks: procedural, predicate, operator, space of states, hypergraph, logistical model, mixed representations.
10. Use of scenarios, logistical models of tasks, and reasoning about tasks (by graphs).

Although the term ontology was never used by Krupa, his framework evidently meets common definitions of ontology, i.e. it is a formal explicit description of concepts in a particular domain. It is descriptive in the Seidewitz's sense (2003), and also prescriptive. It is interesting to note that the dyadic construct of tasks and resources

[4] Krupa has mostly published in research reports of limited circulation. His contributions were summarized in the later of refereed publications.

introduced by Krupa preceded the SOA paradigm, as services and tasks are dual concepts. Unfortunately this framework was never tooled, nor widely disseminated.

A set of TOVE (Toronto Virtual Enterprise) ontologies was developed to create an enterprise infrastructure in the form of an enterprise ontological model, which would have an ability to deduce answers to queries about the tasks in industrial environments (Fox et al. 1996). The tasks have been specified in great detail and this extends beyond MRP to include logistics, physical distribution and concurrent engineering, i.e. coordination of engineering design. A single company perspective was taken and resulted in a set of ontologies, including: (i) resource ontology (Fadel et al., 1994); (ii) cost ontology (Tham et al., 1994); (iii) organization ontology; it captures structural concepts of the organization such as: goal, division, subdivision, agent, role, and resource (Fox et al., 1996; Fox et al., 1998); (iv) product ontology (Lin et al., 1996); (v) activity-state-time ontology; it represents a foundation for other ontologies and acts as a top-level ontology (Fox and Grüninger, 1998); (vi) ontology for quality management (Kim et al., 1999).

Hellingrath et al. (2009) presented the FRISCO ontology for automotive supply chains. It encompasses five separate models: (i) Sourcing Model includes data about products sold to customers and parts procured from suppliers; (ii) Resource Model contains information on manufacturing resources and their capacities; (iii) Adjustment Measure Model maintains structures used to represent network adaptability; (iv) Demand Constraint Model describes relations between demand and capacity to allow real-time responsiveness; (v) Time Model provides structures for different calendars in order to create common understanding of dates between customers and companies.

Al-Jumaili et al. (2012) investigated possibility of using ontologies in the context of eMaintenance aiming at easier exchange and better quality of the maintenance related data.

Giménez et al. (2006, and 2008) proposed ontology for complex product modeling, which was expected to provide foundations for a distributed product data management supported by Semantic Web technology. The ontology suggests three abstraction levels for representing product-related concepts: product family, variant family and product (or physical item). A common formal vocabulary concerning product data that formalizes both processes of information aggregation and disaggregation that occurs during production planning activities is a particular focus of this approach.

Zdravković and Trajanović (2009) described approach to product ontology built upon the concepts of product topology and product function modeling. They used so called semantic information pool for manufacturing supply networks. The aim is to decrease human intervention in product data exchange, as well as to add value through recognition of design intent by automated referencing to manufacturing competences. The ontology includes design representation and functional product representation, based on common product classification schemes.

Borgo and Leitão (2004) described the ADACOR (ADAptive holonic COntrol aRchitecture for distributed manufacturing systems) ontology using the DOLCE methodology. The alignment of ADACOR with DOLCE resulted in a provision of a well formalized and founded ontology.

Alsafi and Vyatkin (2010) proposed an approach to achieving fast reconfiguration of modular manufacturing systems, based on an ontology-based reconfiguration agent. The agent uses ontological knowledge of the manufacturing environment for the purpose of reconfiguration without human intervention. It infers facts about the manufacturing environment from the ontological knowledge model and then decides whether the current environment can support the given manufacturing requirements. The knowledge model is based on the previously mentioned MASON ontology.

Lanz et al. (2006) proposed an approach to share platform independent product-process knowledge between the assembly process and system design, including simulation of manufacturing system. An ontology based on feature-based product-process-system model is used. Product knowledge includes geometric and non-geometric information. It addresses various aspects, e.g.: functionality, options and variants, etc. in order to represent product structure; rules, constraints and assembly-specific information in relation to the product model. In the design and modeling of assembly process, features form foundation for analysis and knowledge acquisition of the product. Furthermore, process, system and visualization sub-ontologies are roughly presented.

Pawlaszczyk et al. (2004) introduced an enterprise ontology to optimize inter-organizational and distributed co-operations. It is distinctively tailored to the mass customization environment and enables modelling of different scenarios concerning development or implementation of mass customization.

Scheuermann and Hoxha (2012) proposed intelligent supply chain management (SCM) based on a combination of Semantic Web technologies and SOA. They introduced dedicated ontology to semantically annotate logistics services using a three-layered model: (1) logistics ontologies providing foundation for defining formal semantics of consensual logistics knowledge; (2) semantic logistics service descriptions used for representation of atomic logistics services for description of service features and utilization of logistics ontologies of Layer 1 for semantic annotation; (3) atomic logistics services composed into complex logistics processes. This model includes elements to describe both, declarative and procedural aspects.

Fayez et al. (2005) proposed an OWL representation of the SCOR model for supply chain simulation. The ontology captures the distributed knowledge being required to integrate several supply chain views in order to support the construction of simulation models. A further study of representation of the SCOR model by means of ontologies is presented by Zdravković et al. (2011).

Daniele and Pires Ferreira (2013) described core ontology for logistics focusing on the concept of physical resource. A limited taxonomy of resources and their structures and some axioms in reference to relations between resources have been proposed.

Chandra and Tumanyan (2007) applied an ontology to systematically record knowledge about organizational and problem-specific issues for SCM. They proposed an information modelling framework to create a taxonomy of supply chain problems and operations to alleviate operational uncertainty.

Leukel and Kirn (2008) developed a logistics ontology based on the SCOR (Supply Chain Operations Reference) model to capture core concepts of inter-organizational logistics. The proposal facilitates description of activities in logistics and provides relations and attributes.

Haugen and McCarthy (2000) proposed an extension of the REA (Resource-Event-Agent) Ontology, which was originally designed for the accounting domain, to support Internet-based supply chain collaboration.

Ye et al. (2008) proposed a supply chain ontology to enable semantic integration between heterogeneous supply chain information systems. The supply chain setting is a web-based one or a virtual enterprise, with no specific industry focus, and it is not the closed supply chain system where partners have already reached the agreement on the vocabulary to be used. In such a setting supply chain partnerships are created dynamically and may last only for a short period. The ontology consists of the following top level classes: (i) supply chain; (ii) supply chain structure; (iii) party; (iv) role; (v) purpose; (vi) activity, (vii) resource, (vii) transfer object, (vii) performance and performance metric. It was coded in OWL to enable Web-based supply chain integration. The ontology uses the skeletal method to capture concepts and relationships of the domain.

Engel et al. (2014) proposed an ontology-based, knowledge-assisted platform to collaboratively create, adapt and steer supply chain networks. Such a platform should allow to reuse domain knowledge captured in previous supply chain projects and support simulation of various network configuration.

Mettler (2010) presented a formal ontology containing concepts useful to analyze manufacturing networks as service systems. The ontology consists of sixteen key constructs like business areas, functions, roles, partners, goals, success factors, performance indicators, incentives, various resources and processes. For every construct exemplary instances or sub-classes, and the relations of them with other constructs, are defined.

Sandkuhl et al. (2013) investigated integration of information systems and production planning systems in enterprises with physical systems, like automation and control systems, into Cyber-Physical Systems (CPS), with focus on the logistics domain and on a service-oriented approach. The core proposal is a generic architecture for Logistics-as-a-Service systems, representing elements of the logistics network as services. It is enriched with concepts from competence management and ontology matching. Ontology-based competence profiles are proposed for representing individual and organizational competences. Ontology matching contributes to configuring and finding resources in LaaS. Within the ontological representation of services and competences, multi-lingual ontology matching was also proposed.

Brock et al. (2009) discussed application of semantic modeling to allow free flow of models that are used along planning and control within a logistical network. Their approach is intended to improve the productivity of logistical modeling in reference to operations management and control.

Most of the reviewed publications provide a rather limited description of ontologies and remain abstract. Many of them address some narrowly focused or specific aspects or are limited to very few abstract concepts. Other apply ontology languages upon some existing models. Such narrow focuses are somehow understandable, as papers are typically of limited size. But after closer analysis of details in these publications it is unquestionable that most authors do not target anything else but a rough vision of the ontology.

Most papers lack rich formal semantics. Description logic is rarely used. Limited taxonomies usually lack formal axioms. Decision making aspect, i.e. with regard to the planning and control, is mostly not discussed. The service-oriented paradigm is ignored or even neglected. The potential of semiotic interpretation of resources, tasks and other classes is rarely explored. Most publications bypass important but difficult aspects of the domain, e.g. details of planning and controlling operations, dynamics and behavior of the system etc. The heterarchical paradigm is also rarely addressed. Under-specification is never incorporated into discussed conceptualizations.

The most important research gap identified in reference to the dynamic behavior within the domain, refers to the complexities and discrepancies that may arise along planning, controlling or execution of operations. Among them the following are typical: correlations, interdependencies, synchronizations, static and dynamical (temporary) fits and conflicts, blockings, starvations etc. The roots of them are analyzed by the literature to a very limited extent, if at all, like e.g.: layout driven limitations to flows; dynamic transformations of temporarily coupled resources and tasks (orders, flows etc.). Only one of the reviewed papers considers distinction of enduring and perduring classes of resources and tasks.

On the other hand the reviewed literature directly or indirectly exhibits the enabling potential of ontologies. However, it is mostly understood narrowly, i.e. in reference to direct advantages of ontologies, like provision of better performance, improved changeability, and other gains from utilization of knowledge or new functionalities. Systemic advantages that may lead to novel solutions in terms of system architectures, processes and controls are rarely considered, and if at all than not in-depth.

Following the conclusions from literature review there are good reasons to argue that the research concerning ontology-aided operations management for manufacturing and logistics is not as advanced, as – to compare - in the field of medical informatics. The conclusions fully justify the following recommendations for further research concerning the discussed domain:

1. Development of core ontology for manufacturing and logistics operations management that could facilitate domain, sub-domain or application ontologies.
2. Development of ontologies equipped with spatiotemporal and mereotopological transformational abstractions to represent dynamic and spatial complexities arising along operations management, and exploiting advantages of the SOA paradigm.
3. Development of new conceptualizations for various and diverse structures of resources (from systems to collectivities), processes (aiming changeability), and planning and control of operations (using alternative and novel control structures and rules, like heterarchical, distributed, herd or local controls).
4. Addressing cross-organizational operations management (i.e. beyond MRP/ERP).
5. Exploiting advantages of localized, globalized or outsourced (public) intelligence, and also the potential of merging ontologies and Big Data capacities.
6. Adapting current mode of operations management to local and temporary factors.

4 The eScop approach

The primary focus of the eScop approach is on re-configurability of production systems. The shop floor control (SFC) level is addressed with regard to highly automated manufacturing systems, namely Manufacturing Execution Systems (MES) and deterministic real-time control. The main declared eScop objective is to allow supervisory and command capability of production systems in an open manner for easy system re-configuration and "plug & produce" adding mode of new equipment. The wanted capability supposes the hot-plug of new equipment and additionally adaptation to the removal of equipment also at a runtime.

Re-configurability is an instance of wider concept, i.e. of changeability (Wiendahl et al., 2007). Changeability is commonly being referred to various changes, ranging from limited modifications of given system, through its reconfigurations up to major evolutionary transformations. Depending on the context the term may be represented by various characteristics, like: flexibility, agility, transformability, re-configurability, modularity et al. Changeability is also referred to various items, ranging from system and resources, through products and services, to processes and operations. It is recognized as an important asset within different domains.

The eScop targets to overcome major existing shortcomings of the SFC level, i.e. the MES systems for automated manufacturing. The novelty of approach is rooted in simplicity and flexibility, but not in sophisticated standardization This contrasts eScop to the existing approaches, e.g. to the widely promoted ISO/IEC 7498-1 standard for Open Systems Interconnection (OSI) model, or to the many standard / reference architectures for integration of manufacturing systems, which are reviewed in section 6 of the first chapter of this book. The eScop architecture comprises three layers (Fig. 2):

- Physical Layer (PhL),
- Representation Layer (RPL) and
- Orchestration Layer (ORL).

The goal of eScop approach is achieved by introducing an innovative combination of particular novel technologies and abstractions to be used by all the layers. They are:

- embedded systems used as Remote Terminal Units (RTUs) by the PhL;
- ontological knowledge representation about manufacturing systems in the RPL;
- SOA technology for orchestrating services used by the ORL.

The Physical Layer includes service-enabled embedded devices performing deterministic real-time control together with operators, who are equipped with embedded appliances. Some functions are outsourced to the RPL, namely:

- device functionality description, to allow device integration into overall system,
- device interface description, to allow interactions with the device,
- device visualization description to allow customized "look & feel" Human-Machine Interface (HMI).

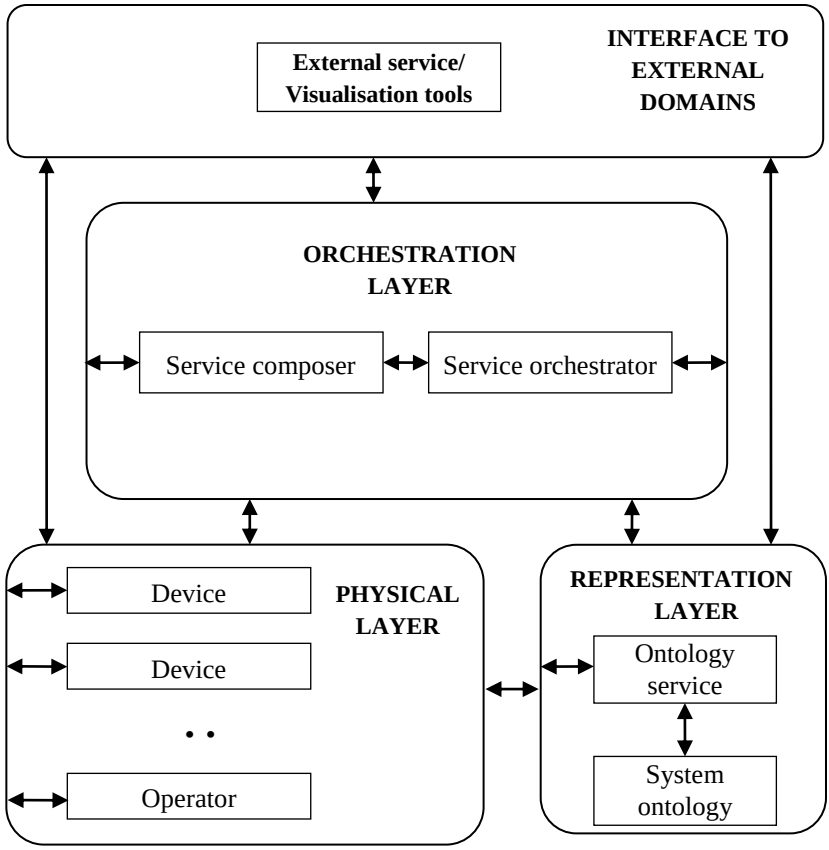


Fig. 2. Basic view of the eScop architecture (based on the eScop project deliverables)

The embedded devices are expected to dynamically update the knowledge model of a system that is handled by the RPL. Hence the knowledge base reflects current status of a system and allows components at orchestration and physical layer to make knowledge base updates and queries along the decision making. The RPL utilizes ontology language to support system modelling, integration and visualization. Currently it is the OWL. The RPL can be also used to facilitate simulations of the production system. The Orchestration Layer provides a supervisory control over the system.

The ontology of a manufacturing system according to the eScop approach incorporates three sub-domains, i.e.: physical, technological and control domain.

The physical ontology includes two subclasses, i.e. subsystem class and components class. The first one is subdivided into different sub-classes, e.g. department, measurement station. The component subclass includes the following lower level subclasses: operator, processor, storage, transporter, container, RTU, sensor, tool, fixture. All of them are again divided into subclasses.

The technological domain represents process aspect of manufacturing system ontology. It includes process routings, transportation routings and operations as the key subclasses.

The control domain describes how planning and control of the system are carried out. It incorporates two key subclasses, i.e. controller and rule. The controller may be a person or a software agent that executes planning and control functions, being in charge of decision making. All the above listed classes are described in terms of properties and other characteristics. When used by a particular application they have to be populated by instances and individuals.

The ontology is supplemented by ontology service. It embraces internal functions of connecting and querying the knowledge base using SPARQL queries, including updates, as well as dedicated interfacing functions. The latter enable interactions and service performance for different elements of other layers capable to use the ontology service, like operators, devices, software agents etc. One of them is the reasoning module, which facilitates inference of implicit concepts or execution of rules defined in the ontology. All together the ontology service provides access to the stored knowledge and enables ontology based reasoning.

The architectural concept of eScop supported by the use of relevant technologies and concepts, particularly of ontology engineering, is expected to provide distinctive features. Among them the following are of major importance:

- merged power of ontologies and SOA-based control approaches enables interoperability between different components thanks to the common communication interface, hence realizing a fully open automated manufacturing environment,
- enhanced human-machine and machine-machine interactions, including interface auto-configuration depending on targeted user needs,
- provision of a reference architecture made of kernel modules capable to connect many service applications in order to allow supervisory and command capability in a new open manner,
- provision of services suitable for different application domains, hence allowing smooth integration and reuse of independently developed software components and embedded appliances, integrating different software tools,
- cross-sectorial reusability of architecture platforms and embedded appliances,
- capability to support flexible re-configuration and knowledge update of newly plugged or unplugged equipment,
- standardization of equipment interfacing which allow to reduce commissioning effort and ramp-up time, whilst enhancing context-awareness, maintainability, modularity, re-usability, safety and versatility of production systems.

The above features provide significant advantages, like reduction of costs and development cycles of the system design, reduction of effort and time required for revalidation and recertification of systems after making changes, increased capability to manage complexity.

5 Extendability of the eScop approach

This section investigates possible extensions and adaptations of the eScop approach. The insights from industrial needs and requirements together with literature review were used as the key inputs for conceptual research. A relevant use of ICT and AT technologies was also considered to facilitate the novel capacities and functionalities. All identified and conceptualized ideations are briefly presented herein.

Although the original focus of eScop approach was on automated manufacturing systems, it actually addresses various environments of interoperating human and technical elements, equipped with smart appliances. Also understanding of reconfigurability is not restricted to structures of systems considered as lasting static settings. It can be easily extended to address spatiotemporal transformations and reconfigurations of resources, including the flowing items. Similarly tasks, with regard to their spatiotemporality, are subject to transformations, including reconfigurations.

The eScop approach makes possible and affordable integration of large networks of users and items, all equipped with or supported by embedded smart appliances. The items may be heterogeneous devices and facilities, as well as stored or flowing material and information items. This is made possible by utilization of different protocols developed by the World Wide Web Consortium (W3C), like those for Web services.

The above means that eScop architecture may be extended to various heterogeneous configurations. They may range from local proprietary networks to open global networks integrated by the means of Semantic Web. All of them may be eventually supported by public or proprietary cloud-services or other Web services. The grounding frame for such configurations is schemed below (Fig. 3).

The enabling potential of eScop approach is rooted in particular capacities of its architecture, assuming it is merged with some frontier technologies, primarily with the SOA and Web services, and secondly with the means of ontology engineering. The following features seem to play a generic enabling role:

1. Interoperability and openness: (i) connectivity-related openness, i.e. by different protocols; (ii) communication openness, as direct communication is possible between devices and other items (e.g. machine-to-schedule, truck-to-depot) together with the use of Web based interfaces (appliances and users are able to access Internet from any device or HMI, like: laptop, tablet, phone etc.); (iii) application openness (as the SOA paradigm is respected); (iv) systemic openness, i.e. integrability of systems, flows and processes with regard to flexible semantic representation of various items; (v) complexity-related openness, i.e. ability to represent and handle semantically represented spatiotemporal structures, considering mereotopological expressions, like e.g.: demand/task structures (projects, orders, etc.), interdependencies concerning coordination patterns (e.g. synchronizations, time-rigidness etc.).
2. New and mixed forms of networking (and followingly of organizational structures), including: peer-to-peer, centralized hierarchical, decentralized, non-hierarchical, segmented, herds and other types of systems and collectivities.

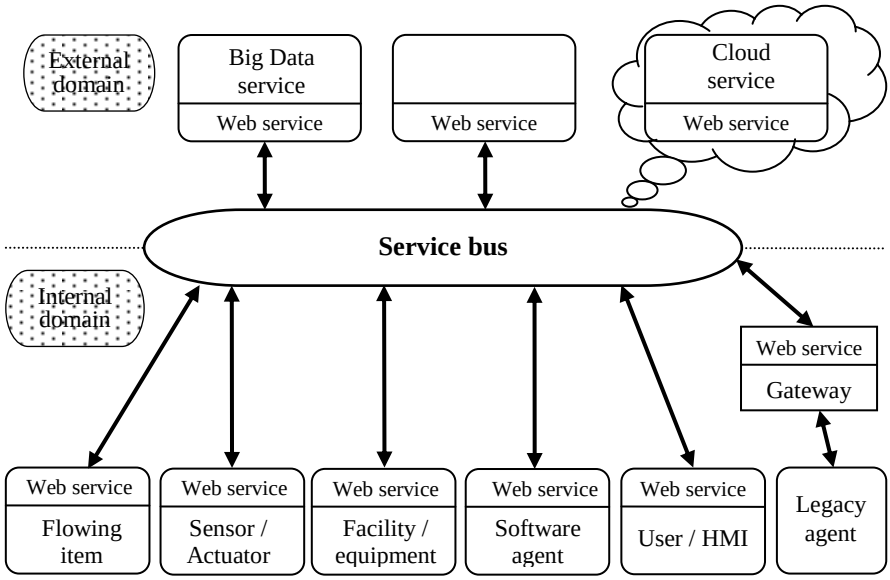


Fig. 3. Perspective on extended usability of the eScop approach

3. Knowledge-aided smartness or intelligence: granulated or distributed, localized or externalized (e.g. supported by external Cloud and Big Data services), extendable and upgradeable, capable to be built in into the applications in a run-time mode. Adaptiveness, self-diagnostic and enhanced diagnostic can be facilitated this way, including early-warning, as well as smart/intelligent decision making. Also due to possibility of tacit knowledge representation and some learning capacities.
4. New and mixed forms of management and control of processes: distributed / non-hierarchical / externalized / localized planning and control, ranging from self-management of subsystems or devices or orders, through collective distributed self-coordination (e.g. like self-management of devices or orders as herds, mimicking the herd behavior), up to externalized and/or centralized planning and control., using various mechanisms, possibly following the idea of pull-flows, through bargaining, auctioning and similar mechanisms;
5. New forms of encapsulation (i.e. other than data encapsulation as implemented in the Internet) and virtualization of spatiotemporal structures and entities, like e.g. of: demands, processes, flows, stocked items, etc. As it is illustrated later on, these capacities may drive novel forms of economic institutions, new types of infrastructures and so on.

Exploitation of the above features may eventually provide some emerging capacities, like new functionalities.

Consequently enormous opportunities but also challenges may affect the manufacturing and logistics area, when the above features are exploited. If new management and control technologies are facilitated by some frontier ICT/AT provisions, new

ideas concerning structures and processes could emerge, going beyond current mainstream paradigms.

The following ideations for ontology-aided manufacturing and logistics have been conceptualized, using inspirations from the all preceding discussions. All of them can lead to significant advantages in relation to the existing solutions. All of them, despite some requirements for further developments, could be realistically supported by some sound technologies. The following the novel ideations of systemic solutions and functionalities have been conceptualized (some of them overlap, which is not an issue, while a better transparence and illustration is achieved with this respect):

1. *Alignment of electronic market with a provider of 5PL (5th Party Logistics)*: Electronic market can be integrated through Web services (possibly a Cloud service) and by logistics providers (or logistics infrastructure, like the Physical Internet - π), with a pool of manufacturers. Different coordinating mechanism can be exploited. Demands and material flows could be encapsulated along operations, thus thriving from transaction economy of scale[5]. Eventually the electronic market could be integrated with an active marketing solution run by the social Internet.
2. *Alignment of supply chain*: Common coordination could be used to integrate management and control of demand and material flows within vertically integrated production network (supply chain), or alternatively in an inter-organizational network (supply chain). Various coordination mechanisms can be exploited, depending on the characteristics of demand, products and processes, and particularly in reference to technical and other couplings. The used coordination modes can range from centralized planning and scheduling to local adaptive controls. Demands and flows of goods could be eventually encapsulated along the operations (e.g. pooled or split orders), thus thriving from transaction economy of scale or economic order quantities. Distortion of demand could be effectively avoided using in a relevant way such solutions, hence turbulent flows tamed. Also utilization of resources could be significantly improved this way. This solution could be applied as a dedicated service or public (cloud) service, also for corporate supply chains.
3. *Distributed and/or heterarchical self-management of operations within a manufacturing system*: A cell or an installation or another facility could be operated basing on smart appliances embedded or assigned to various components of the system. Manufacturing and logistical orders would be actually managed and controlled by intelligent agents, who would solely supervise their execution. The agents would apply for resources, including a run-time mode, with regard to completion of consecutive operations. Similarly resources would be equipped with managing agents, who would apply for operations. The workflows would be limited by technological

[5] This ideation is somehow preceded by the solution of Uber Technologies, which presumes new way of transportation, as yet in agglomerations. It is called 'urban logistics fabric' and assumes occasional involvement of drivers, who controlled by a smartphone can shuttle passengers together with picked up items (food, grocery, packages) along their way. A similar solution is being developed by Amazon: a mobile application that would ad hoc employ individuals rather than carriers, to drop off packages en route to other destinations. Not to forget, a similar way of services was supported by the logistical infrastructure of Incas nation.

routings while transportation processes could be composed at a run-time. A range of relevant control mechanisms could be used by both types of agents, for example auctioning or prioritizing mechanism. Hence a competition for resources or orders could be resolved some way. These mechanisms could consider some spatiotemporal conditioning factors, adapting the control of operations to the current situation or context, i.e. like following state transitions of the system. The agents could be supervised by operators and other users. The above framework may eventually exhibit self-management of resources and tasks. This framework is distinctively different from the hierarchical management and control that is characteristic for the existing ERP and SFC / MES systems.

4. *Externalized or outsourced management and control of operations within a system*: A cell or an installation or another manufacturing facility can be managed and controlled by an internally (proprietary, or intranet) or externally (e.g. Cloud service) outsourced Web-service. Extended and enhanced functionalities and performance can be supported this way, e.g. due to exploitation of Big Data services. Possibly novel operational processes can be enabled this way, including both, the management and control processes.
5. *Additional and more efficient systemic solutions and functionalities supported by Web services*: Due to advantages of Cloud computing and Big Data technologies new functionalities could be added for occasional or run-time use. Alternatively those existing could be provided in a more efficient or a more functional way, e.g. by exploitation of Big Data received in a mass scale, not just from a one company. Examples could be: monitoring of resources, processes, and quality, process planning, setting time standards et al.
6. *Herd control of operations*: Complex manufacturing demands could be distributed to various providers or vendors, then operating like a herd. Management and controls could be eventually developed as bio-mimetic and eco-mimetic imitations, e.g. hormonal control. This ideation is distinctively different from the concept of distributed management of operations.
7. *Adoption to KPIs (Key Performance Indicators) and state / phase or contextual transitions*: Management and control of operations could be adapted according to ongoing state or phase transitions of systems and processes or their environment, and taking into account the targets set in terms of KPIs. E.g. dispatching rules in a manufacturing system could be changed according to temporary bottlenecks, processes (routings) could be redefined in case of breakdowns, outages and jams, etc. This kind of functionalities is primarily enabled by an intelligent control. However it must be underlined, that they can be significantly enhanced when exploited within extended environments, like e.g. supply chains. Thus the capacity to anticipate the phase transitions could be facilitated in a much more effective way.
8. *Distant and 'cross-border' coordination or control*: Sometimes significant advantages can be thrived from coordination of distant operations or processes, including real-time controls. This may apply to a large plant or facility, or alternatively to a globally distributed process, i.e. operated at distant facilities. Integrated collection of sensors and actuators or appliances that principally operate inside localized systems can provide various opportunities. E.g.: 'global' optimization of a

complex continuous process in a plant; coordination of technical parameters concerning outcomes of some coupled orders, crossing organizational boundaries.

The above ideations illustrate a range of novel conceptualizations for manufacturing and logistics operations, which are enabled by particular extensions or adaptations of the eScop approach, by reflecting on its grounding triad, i.e.: 'embedded smartness'–'ontology-based representation'–'SOA supported interoperability'. Various knowledge-driven distributed, non-hierarchical, self-managing and cross-organizational solutions can be implemented by merging ontologies with Web-services and using SOA-based architectures.

The novel operational and management processes, organizational structures and functionalities can provide significant performance advantages and may outcome in new capacities in terms of business competence.

It must be underlined that although the proposed ideations are visibly rooted in the eScop approach, they go far beyond its original framework. Therefore they set directions for further research or practice oriented developments.

6 Conclusions

It is advocated in this chapter, that developments of semantic technologies and distributed systems following the Service Oriented Architecture achieved a critical mass and acceptable level of maturity, to be widely adopted by the manufacturing and logistics area. The conclusions from empirical research confirm crucial role of ontologies, as seen from the perspective of industrial practice. The presented ideations, which are based on various extensions and adoptions of the eScop framework, can provide significant advantages, including:

1. Use of shared and extendable knowledge in distributed environments, including the run-time mode; this can facilitate distributed intelligence used to support improved decision making, self-diagnosing and adaptability.
2. Improved openness, interoperability, integrability, convertibility provide increased potential for cross-layer, cross-functional, cross domain and cross-process integrations, including intra- and inter-organizational integrations.
3. Better functionality and performance of systems, resources and processes.
4. Significantly enhanced re-configurability, transformability and adaptability of systems, resources and processes.

The above conclusions call for extended effort concerning diffusion of the discussed concepts and ideations. The particular challenging areas of further developments are:

1. novel business processes, to fully exploit broken organizational barriers and all other advantages of ICT/AT innovations;
2. new structures of business systems and resources, thriving from advanced concepts of distributed systems;

3. new concepts of management, emerging from the power of modern management technologies and
4. new management and control paradigms, possibly rooted in the bio- and eco-mimetic imitations.

The relevant developments of fundamentals and technologies of ontology engineering, and of SOA-based (Web-)services, define another important stream of further research efforts.

The eight example ideations, which are conceptualized in the chapter, seem to be significantly important in terms of their novelty and practice related impact, with regard to the extended uses of eScop approach.

Finally, it must be noted that there is a big gap between the needs and requirements of industries, and the literature provisions. Apart of the research gap, as identified by the literature review, the inconsistence of research and practice focuses exists. Therefore, the needs and requirements of industries, which are very precise, should be added to the research agenda.

Acknowledgement: This work has been co-funded by the Grant of Warsaw University of Technology No. PSP 504/01518/1103/40.000105. The author also thanks to Mr. Krzysztof Ejsmont for his valuable comments.

References

Al-Jumaili, M., Wandt, K., & Karim, R. (2012) eMaintenance Ontologies and Data Production. *Proceedings of the 2nd International Workshop & Congress on eMaintenance*, Luleå, 2012.XII.12-14: 191-196.

Alsafi, Y., & Vyatkin, V. (2010) Ontology-based Reconfiguration Agent for Intelligent Mechatronic Systems in Flexible Manufacturing. *Computer Integrated Manufacturing* 26: 381-391.

Battista, C., Giordano, F., Iannone, R., & Schiraldi, M. (2010). A proposal for a standard framework for simulating and modelling manufacturing systems. In: S. Digiesi, G. Mossa, G. Mummolo, L. Ranieri (eds), *Sustainable Development: Industrial Practice, Education and Research*, Vol. 2: 1-5.

Borgo, S., & Leitão, P. (2004). The role of foundational ontologies in manufacturing domain applications. In: *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, LNCS Vol.3290: 670–688.

Brock, D.L., Schuster, E.W., Allen, S.J., & Kar, P. (2009) An Introduction to Semantic Modeling for Logistical Systems. *Journal of Business Logistics***,** Vol**.** 26/2**:** 97–117.

Cândido, G., & Barata, J.A. (2007) Multiagent Control System for Shop Floor Assembly. *Lecture Notes in Artificial Intelligence*, 4659. Springer: 293-302.

Chandra, C., & Tumanyan, A. (2007) Organization and problem ontology for supply chain information support system. *Data & Knowledge Engineering*, Vol. 61, 2007/V: 263–280.

Daniele, L., & Ferreira Pires, L. (2013) An Ontological Approach to Logistics, In M. Zelm, M. van Sinderen, L. Ferreira Pires, & G. Doumeingts (eds), *Enterprise Interoperability*. John Wiley & Sons: 199-213.

Dassisti, M., Panetto, H., Tursi, A., & De Nicolo, M. (2008) Ontology-Based Model for Production-Control Systems Interoperability, *Proceedings of the 5th CIRP International Conference on Digital Enterprise Technology*.

Engel, T., Venkatesh, V., Goswami, S., Krcmar, H., & Bhat, M. (2014) An Ontology-based Platform to Collaboratively Manage Supply Chains. *Proceedings of International Conference on Production and Operations Management (POMS'2014)*, Production and Operations Management Society, Atlanta, 2014/VIII: 1-10.

Fadel, F.G., Fox, M.S., and Grüninger, M. (1994). A generic enterprise resource ontology. In: *Proceedings of the 3rd IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'94)*: 117–128.

Fayez, F., Rabelo, L., & Mollaghasemi, M. (2005) Ontologies for supply chain simulation modeling. *Proceedings of the 37th Conference on Winter simulation (WSC'05)*, 2005/XII: 2364–2370.

Fox, M.S., Barbuceanu, M., Grüninger, M. (1996). An organization ontology for enterprise modelling: preliminary concepts for linking structure and behavior. *Computers in Industry*, Vol.29/1–2: 123–134.

Fox, M.S., Barbuceanu, M., Grüninger, M., Lin, J. (1998). An organization ontology for enterprise modelling, In: M. Prietula, K. Carley, and L. Gasser (Eds.), *Simulating Organizations: Computational Models of Institutions and Groups*, Menlo Park: AAAI/MIT Press: 131–152.

Fox, M.S., and Grüninger, M. (1998). Enterprise modeling, *AI Magazine*, 1998/Fall: 109–121.

Garetti, M., & Fumagalli, L. (2012) P-PSO ontology for manufacturing systems, *Proceedings of the 14th IFAC Symposium on Information Control Problems in Manufacturing*: 247-254.

Giménez, D., Vegetti, M., Henning, G., & Leone, H. (2006) PRoduct ONTOlogy. Defining product-related concepts for production planning activities. In W. Marquardt, C. Pantelides (eds), *Proc. of the 16th European Symposium on Computer Aided Process Engineering and 9th Int'l Symposium on Process Systems Engineering*, Elsevier 2006: 2219-2225.

Giménez, D., Vegetti, M., Leone, H., & Henning, G. (2008) PRoduct ONTOlogy. Defining product-related concepts for logistics planning activities. *Computers in Industry*, Vol. 59/2-3: 231-241.

Grubic, T. & Fan, I.-S. (2010). Supply chain ontology: Review, analysis and synthesis. *Computers in Industry*, 61: 776–786

Haugen, R., & McCarthy, W.E. (2000). REA - a Semantic model for Internet Supply Chain Collaboration. In: *Online-Proceedings of the ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications - Business Objects and Component Design and Implementation Workshop VI: Enterprise Application Integration*. 2000/X. http://www.jeffsutherland.org/oopsla2000/mccarthy/mccarthy.htm accessed 2014.IX.22.

Hellingrath, B., Witthaut, M., Böhle, C., & Brügger, S. (2009). An Organizational Knowledge for Automotive Supply Chains. In: *HoloMAS 2009, LNAI*, Vol.5696: 37-46.

IEC 62264 (2002). Enterprise-control system integration, Part 1: Models and terminology.

Kim, H.M., Fox, M.S., Grüninger, M. (1999). An ontology for quality management-enabling quality problem identification and tracking. BT Technology Journal, Vol.17/4: 131–140.

Krupa, T. (1984). Method of Task model (in Polish). *Proceedings of the International Conference on Production Systems (SYPRO'84)*, Warsaw, 1984.

Krupa, T. (2006). Elements of organization: resources and tasks (in Polish). WNT, 2006.

Lanz, M., Garcia, F., Kallela, T., & Tuokko, R. (2008) Product-Process Ontology for Managing Assembly Specific Knowledge Between Product Design and Assembly System Simulation. In *Micro-Assembly Technologies and Applications*, IFIP, Vol 260, 2008: 99-108.

Lemaignan, S., Siadata, A., Dantan, J.-Y., & Semenenko, A. (2006) Mason: A Proposal for an Ontology of Manufacturing Domain. In: *Proceedings of the IEEE Workshop on Distributed Intelligent Systems*: 195-200.

Leukel, J., & Kirn, S. (2008). A Supply Chain Management Approach to Logistics Ontologies in Information Systems. *Proceedings of the 11th International Conference on Business Information Systems (BIS08)*, Springer LNBIP, 2008/V: 95-105.

Lin, J., Fox, M.S., Bilgic, T. (1996). A requirement ontology for engineering design. In: *Proceedings of the 3rd International Conference on Concurrent Engineering*: 343–351.

Lin, H.K., & Harding, J.A. (2007). A manufacturing system engineering ontology model on the semantic web for inter-enterprise collaboration. *Computers in Industry*, Vol. 58/VII: 428-437.

Lin, H.K., Harding, J.A., Shahbaz, M. (2004). Manufacturing system engineering ontology for semantic interoperability across extended project teams, *International Journal of Production Research*, Vol.42/24: 5099–5118.

Madni, A.L., Lin, W., & Madni, C.C. (2001). IDEON TM: an extensible ontology for designing, integrating and managing collaborative distributed enterprises. *Systems Engineering*, Vol. 4/II: 35–48.

Mettler, T. (2010). Understanding Manufacturing Networks as Service Systems: An Ontological Approach. *Proceedings of the International Conference on Advances in Production Management Systems (APMS'2010)*, Cernobbio, 2010.

Obitko, M., Vrba, P., Mařik, V., Radakovič, M., & Kadera, P. (2010). Applications of Semantics in Agent-Based Manufacturing Systems. *Informatica*, Vol. 34: 315–330.

Pawlaszczyk, D., Dietrich, A.J., Timm, I.J., Otto, S., & Kirn, S. (2004). Ontologies Supporting Cooperation in Mass Customization - A Pragmatic Approach. *International Conference on Mass Customization and Personalization – Theory and Practice in Central Europe*, 2004/IV: 1-19.

Sandkuhl, K., Lin, F., Shilov, N., Smirnov, A., Tarasov, V., & Krizhanovsky, A. (2013) Logistics-as-a-Service: Ontology-Based Architecture and Approach. *Revista Investigacion Operacional* Vol. 34/3: 188-194.

Scheuermann, A., & Hoxha, J. (2012). Ontologies for Intelligent Provision of Logistics Services. *The Seventh International Conference on Internet and Web Applications and Services (ICIW 2012)*: 106-111.

Seidewitz, E. (2003). What models mean, *IEEE Transactions on Software Engineering*, Vol. 20/5: 26-32.

Soares, A.L., Azevedo, A.L., & De Sousa, J.P. (2000). Distributed planning and control systems for the virtual enterprise: organizational requirements and development life-cycle. *Journal of Intelligent Manufacturing*, Vol. 11/6: 253–270.

Tham, K.D. Fox, M.S., Grüninger, M. (1994). A cost ontology for enterprise modelling, In: *Proceedings of the 3rd IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'94)*: 111–117.

Uschold, M., King, M., Moralee, S., Zorgios, Y (1998). The Enterprise Ontology. *The Knowledge Engineering Review*, Vol.13/1: 31–89.

Wiendahl, H.P., ElMaraghy, H.A., Nyhuis, P., Zäh, M., Wiendahl, H.H., Duffie, N., & Brieke, M. (2007). Changeable Manufacturing - Classification, Design and Operation. *CIRP Annals*, Vol. 56/2: 783-809.

Ye, Y., Yang, D., Jiang, Z., & Tong, L. (2008). An Ontology-based Architecture for Implementing Semantic Integration of Supply Chain Management. *International Journal of Computer Integrated Manufacturing*, Vol. 21/1: 1–18.

Zdravković, M., & Trajanović, M. (2009). Integrated Product Ontologies for Inter-Organizational Networks. *ComSIS*, Vol. 6/2: 29-46.

Zdravković, M., Trajanović, M., & Panetto, H. (2011). Local Ontologies for Semantic Interoperability in Supply Chain Networks. *Proceedings of the 13th International Conference on Enterprise Information Systems (ICEIS'2011)*, Beijing, pp. 22-31.

Designing of Business Models for ICT Products

Robert Wojtachnik

Warsaw University of Technology, Warsaw, Poland

wojtaro@wp.pl

Abstract. This chapter discusses problems related to designing of business models. Contrary to visualisation of business models, designing is a process which focuses on the evolution of such a model. The created model is then analysed using value proposition reconstruction techniques, studies of possibilities of obtaining non-customers, as well as the identification of their needs. Results of the analysis are then used as a basis for the development of prototypes, in which five schemas are adopted. The designed prototypes are evaluated to assess the usefulness for the buyers, evaluate the effective execution of the profit side and possible obstacles. The process ends in planning of subsequent steps.

1 Introduction

As regards innovation classification, we may distinguish conservative and breakthrough innovations. The characteristic features of innovations were presented in Table 1. Examples of Xerox and Microsoft point to issues connected with the implementation of innovations which are of a breakthrough nature, which – as happened in the case of Microsoft – ended with abandonment of the project.

Xerox example shows the first photocopier, which was to replace manual copying of texts. Yet it proved to be too expensive for it to be placed in the market. Xerox was sure of its invention and started to seek a model suitable to allow it to place the device on the market. The concept assumed that the device would be leased, with a certain set limit of copies under a leasing instalment price. Once the copy limit has been exceeded, the lessee would have to pay for each copy made. The new concept was accepted by the market (Osterwalder, 2014).

Table 1. Comparison of features of a conservative and breakthrough innovation (based on Christensen and Raynor, 2008)

Conservative innovation	Breakthrough innovation
Manufacturing of an increasingly better product	Offers new values – simpler, more convenient, cheaper products
Targeted at the most demanding customers by offering top quality	Attract less demanding customers
In conservative conditions experienced companies have a clear advantage	In breakthrough conditions the newcomers prevail over experienced ones

From the other site Microsoft browser project wasn't succeed. In 1998 Microsoft bought LinkExchange, a company specialised in distribution of advertisements between particular pages and initiated an experimental system that matched advertisements to the phrases sought. In 2000 the project was closed. In 2002 Google presented the AdWords programme - similar project to Microsoft one. In 2003 Microsoft started again works on a new browser and an advertisement system based on search results. At that time the Google project was already much more advanced. As a result the project of Yahoo and Microsoft was left far behind Google, and the considerable part of advertisers recognised the advantages of this system. In 2008 Google's revenues constituted 75% of the sum generated by advertisements that accompany search results, while Yahoo achieved 20% and Microsoft 4% (Brandt R.L., 2011).

Studies of the available literature do not indicate in a clear way that all innovations of a breakthrough nature have to encounter similar problems. Nevertheless they are more susceptible to issues related with reception on the market or the overcoming the barrier of high costs ensuing from offering new unknown product features, which are not easy to compare. One of the methods adopted to overcome problems with commercialisation of a breakthrough innovation is to find an appropriate business model for it, which would be accepted by the market (Osterwalder, 2004).

P. Kotler and F. T. De Bes (2013) worked out a classification of innovation levels from the most strategic one to the one that is the of most tactical nature:

- Innovations related to the business model
- Process innovations
- Market innovations
- Product and service innovations

Innovations with respect to the business models comprise profound changes in the way of creating value added by a company. This requires a considerable restructuring and the establishment of a new business unit. Apple example shows just such an innovation. At the beginning of the value delivered to customers by Apple was easy to use the phone. As a result of Apple's innovations began to provide the entire platform based on the IOS system, used to multimedia, calling, entertainment and work. As a result of this change, Apple decided to provide access to the source code so that other companies could form applications on the iPhones. This meant changing the business model and the resignation of earning exclusively on mobile phone sale and attract new customers through a large spectrum of possibilities IOS platform. As a result, the number of phones sold was increased, and there were new revenue streams from the sale of applications in the Apple Store and iTunes media platform.

The remaining innovation levels concern indirectly a change in the business model and merely affect one or a few elements of the business model. Technological innovation does not guarantee success in business – the development of a new product should be combined with defining of the business model (Teece, 2010).

Process innovation means changes in the operating methods of a company with respect to logistics, production or sale. To cite another example of the iPhone company, it implemented a process innovation consisting in withdrawal from the rule of selling

products only via a single operator or exclusive distributor in each country. Therefore the company allowed all operators to sell the telephones.

Market innovation means targeting an offer at a new group of buyers, satisfying a new kind of needs or the presence in a new type of purchase or consumption related situations (situation). The marketing of Smartphone by Apple was a response to the need of satisfying a new type of needs – better user mobility thanks to the application of additional programmes and access to the Internet directly on the telephone.

On the other hand, product and service innovations apply to technological changes, new models or expansion of product lines addressed at the same consumer, needs or situations. Each new version of the iPhone is a product innovation, because it differs from the previous one by parameters, increased power and options.

In this chapter, a task of describing the process of creating a business model used for the implementation of the ICT technology, was undertaken. Taking into account the abundance of available models, designing may lead to the creation of several different business models which are possible to implement. Consequently a need arises of assuring an assessment of the developed business models and should it prove to be necessary a need of working out indispensable streamlining changes.

2 Creation process of a business model

Depending on whether the starting point for the creation of a business model is a ready product that requires commercialisation, or whether an innovative product should be created as an element of the process of business model creation, the process is likely to be executed in a slightly different way. This chapter presents the process of creating a business model for an existing product.

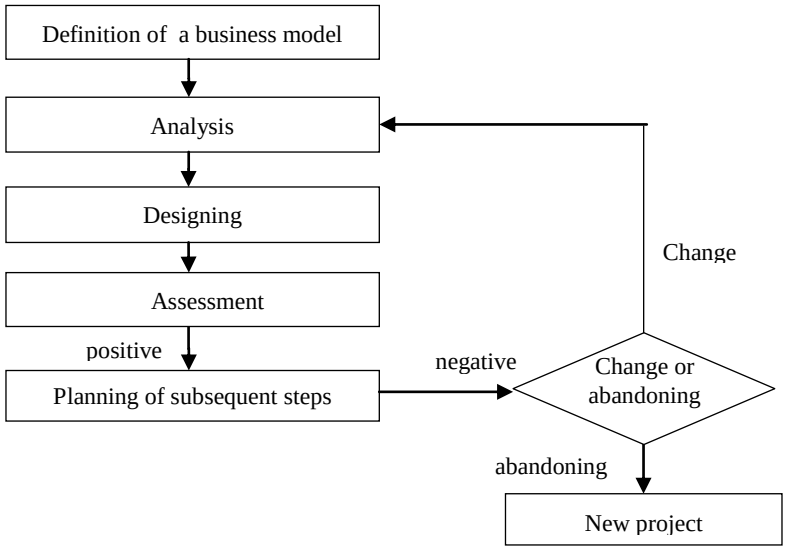


Fig. 1. Process of creating a business model

Fig. 1 presents the process of creating a business model, which will commence the marketing of product. In the next step an analysis of particular elements of the canvas will be made in order to indicate the possibilities of model modification, which is to be performed in the next step based on standardised canvases. The created models are assessed with respect to the potential for success as well as they are compared with the aim of selecting the best solution. If the team is unable to assign a positive assessment to any of them, then the one which shows prospects for improvement is chosen or the whole project is abandoned. A model which has been assessed positively is implemented, and also further activities are planned aimed at its development.

3 Definition of a business model

In order to define a business model use was made of a Business Model Canvass, which is to allow the visualisation of nine key elements: customer segments, value proposition, relations with the customers, channels, revenue streams, key resources, key activities, key partners, and the cost structure. The described model is not the only feasible and is not a final project. It should be considered as a description of the existing condition or an idea about the way in which the product can be commercialised.

4 Analysis

The analysis stage is aimed at researching the topic and accumulating knowledge concerning customers, technologies and the surroundings. The adoption of the proposed techniques will help create more innovative business models. This section will also discuss methods of analysing and reconstructing the value proposition, customers and non-customers, as well as the identification of their changes. Those elements constitute the right side of a business model which affects the way in which a client perceives the value.

Also other tools not described in this section may prove to be useful in the analysis. Hence for an analysis of market borders and their reconstruction use may be made of alternative strategic groups, sectors, buyer chains, complementary offers, functional and emotional factors and a time horizon.

Reconstructing the value proposition .
For the needs of reconstructing the value proposition, Kim and Mauborgne have developed a schema of four activities. The said instrument puts into question the strategic logics and the business model of the sector. It is used for reconstructing the value for the buyers and for the development of a new value curve thanks to the following four activities:

1. Elimination – In some situations the customers have ceased to pay attention to certain factors, on which the competition of companies was based. Those factors have either lost some of their value or reduce the value perceived by the buyers. Hence it

is necessary to find the redundant factors, which are taken by the sector for granted and by which the companies compete, and have them eliminated. Elimination of factors means that no capital out-lays would be made in them. If a lot of such factors are diagnosed, the elimination should be based on the costs of investment in those factors.

2. Reduction – In their efforts aimed at satisfying customers' expectations and overcoming the competition, companies strive to outdo one another, as regards properties of products or services. Excessive investments in certain factors cause an increase of costs with simultaneous lack of additional profits. Therefore it is necessary to find factors which may be reduced much below the standards in the sector without threatening the way of perceiving the offer by customers.
3. Strengthening – As each sector offers certain values, it imposes certain compromises on the part of the customers. At this point it is necessary to find such compromises and consider which factors should be strengthened well above sectoral standards in or-der to provide an uncompromising offer at a minimum costs.
4. Creation – The task of this item is to discover absolutely new sources of value for the buyers and to create new demand for them. Consequently it is necessary to create factors not offered by the sector to date and which are to change the price policy in the sector.

Particularly important activities consist in eliminating and creating, which incline companies to change competition factors and make the existing competition rules lose their importance. The result of the value proposal reconstruction may be presented on the diagram: eliminate-reduce-strengthen-create. Planned activities should be applied on the strategy backdrop, which would be analysed in the stage of assessment.

The application of this tool and visualisation of its results enables to find competitors focused exclusively on strengthening and creating, and consequently causing an increase in the cost structure. At the same time it enforces striving for low costs to break competition of value with costs.

Analysis of non-customers.

Customers may analyse the existing customers and seek similar behaviour patterns among non-customers. It is also necessary to seek common traits of what the customers ascribe value to. Contrary to the classical method, it is possible to consider who non-customers of the company are and how they could be reached. For this purpose knowledge about the environment of non-customers is to be studied. Chan Kim and Mauborgne (2010) distinguished three groups of non-customers:

- Not-customers on our market – These are customers who are in a way forced to use the market offer, but at the same time continue looking for something better. If they get a significant value leap they would abandon it and the frequency of their purchases would increase. A good example is the balanced LiteBox diet as an alternative for restaurants.
- Individuals rejecting the market offer – This is a group of individuals who refuse to benefit from what the sector is offering. They have become familiarised with the

sector offer as an option for satisfying their needs and have then rejected it due to excessive price, satisfying needs in a different way, the needs remain not satisfied. A good example of this group of non-customers may be the JCDecalux advertisements on stops as an alternative option for billboards and advertisements inside transport means.

- Individuals who have not taken our offer into account – These are individuals who have never considered the offer of our market as an option they could use. By focusing on common traits of the customers and this layer of non-customers, we may comprehend how we can attract them. A good example of this group of non-customers is the orientation of companies that produce teeth whitening compounds on the sale to dentists without taking into consideration whitening of teeth by patients at home.

Empathy map.

The Empathy map is a tool developed by XPLANE designated to identify the needs of customers. It has been based on the intention of going beyond the demographic characteristics of the target group and ensuring better comprehending of the surroundings, behaviour patterns, feelings and concerns. Development of such a profile will help better understand for which kind of value proposition the customer would be willing to pay. Fig. 2 presents a template of the empathy map, which describes the following:

- What the customer sees - describes what the customer sees in the environment, who his/her acquaintances are, what problems he/she encounters, the kind of market offer encountered on an everyday basis as compared to the total market offer
- What the customer hears - describes the way in which the environment affects the customer on the basis of what his/her acquaintances say, which media channels affect his opinion and who influences his/her opinions

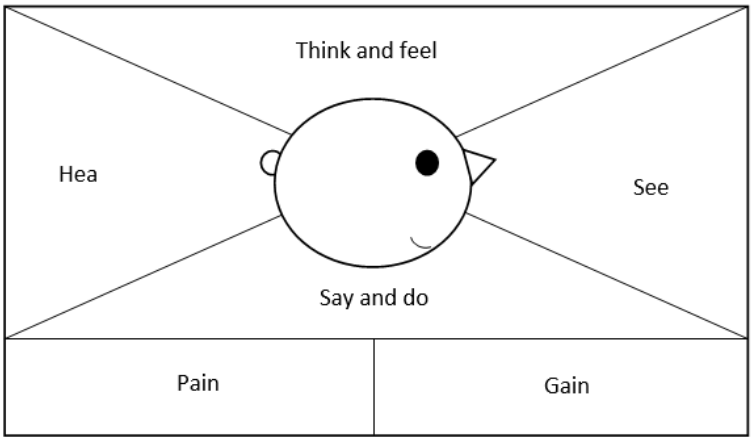


Fig. 2. Empathy map (based on Osterwalder, 2013)

- What the customer thinks and feels - describes what happens in the customer's head, what are the customer's dreams, what worries him/her, what is important for him/her
- What the customer says and does - describes public behaviour of the customer, and namely his/her attitude and what he/she says to the others
- What are his/her concerns- describes what problems are faced by the customer, what obstacles are encountered in his/her efforts at satisfying own needs, and what risk he/she is trying to avoid
- What benefits the customer expects - the customer should consider what the objective that he or she is trying to achieve is and what strategy he/she should adopt to achieve it

5 Designing

The designing stage is important with respect to the creation of innovative business models on the basis of concepts arising from the analysis. The designing will be carried out with the use of five schemas. Those schemas are needed to formulate certain design concepts in the form of archetypes and descriptions which may be used multilaterally. The application of such templates will enable the development of five prototype templates, which will be assessed during the next stage.

Separation.
The concept of separation consists in the distinguishing of three types of activities. In each type of activity different economic and competitive conditions may be observed. The separation of those three activities should be considered to be the optimum solution. Fig. 3 presents the business model canvas divided into three types of operation (Osterwalder and Pigneur, 2010):

- Product innovations – Early development of a product with a unique value for the buyer is connected with lack of reference price (as there is no competition) and the possibility of demanding a higher price and obtaining a considerable share in the market. The company should focus on gaining valuable employees and keeping them, as well as on innovativeness. Of key importance is also the rate of marketisation of the innovation owing to the quickest possible return on investment and outdoing the competition.

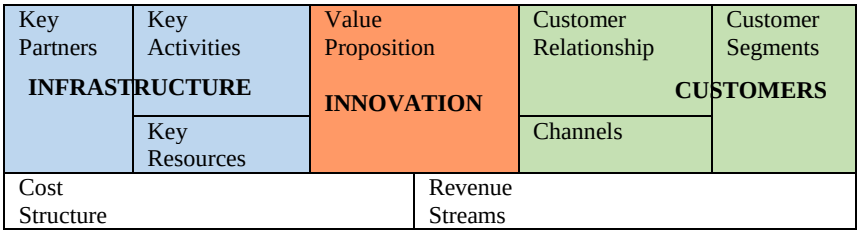


Fig. 3. Separation schema (based on Osterwalder, 2014)

- Managing relations with the customers – Focusing on building relations with the customers causes generation of high costs with regard to obtaining customers. This calls for enforcement of the maximum utilisation of their buying potential. When choosing this strategy, the company should focus on the offered service and think about the customer with this regard. In this strategy economies of scale are of key importance.
- Managing the infrastructure – Focusing on infrastructure owing to high fixed costs imposes a high production level on the enterprise to allow maintaining low unit costs. The company should focus on the costs and strive at standardisation, predictability and performance (Sierpińska and Jachna, 2004). The earnings arise from the economies of scale.

Sectors, in which the separation schema has been applied, comprise the mobile telephony. Product innovations are provided in this sector by minor producers, who attract the appropriate people, and provide such services as navigation, games and music. The infrastructure is provided by manufacturers of the telecommunication equipment or network operators. Mobile telecoms focus exclusively on customer handling.

Long tail.

The model consists in the sale of smaller amounts of a larger number of products or services. The company offer comprises a large number of niche products. Profitability of the model arises from the cumulated sale of a small number of numerous products. This model does not require large outlays on storage owing to insignificant stock levels of products. A condition for the success in implementation of the long tail concept is the development of a platform which would guarantee easy access to niche products. The long tail was popularized by Anderson (Quinion 2011, Anderson 2006), who mentioned Amazon.com, Apple and Yahoo! as examples of businesses applying this strategy.

Given enough choice a large population of customers and negligible stocking and distribution costs, the selection and buying pattern of the population results in the demand across products having a power law distribution or Pareto distribution. It is important to understand why some distributions are normal or long tail distributions. Anderson argues that while quantities such as human height or IQ follow a normal distribution, in scale-free networks with preferential attachments, power law distributions are created, because some nodes are more connected than others (Long Tail, 2015).

The long tail concept has found some ground for application, research, and experimentation. It is a term used in online business, mass media, micro-finance, user-driven innovation and social network mechanisms (e.g. crowdsourcing, crowdcasting, peer-to-peer), economic models, and marketing (Long Tail, 2015). An example of a long tail is establishment of the Lego Factory, where the users could create their own sets of the Lego blocks and order them via the Internet. They had at their disposal thousands of elements and a lot of colour variations.

Multilateral platforms.
These platforms combine two separate but interrelated groups of customers. Multilateral platforms provide value for one customer group provided that there is a possibility of building contacts with the other group. This gives the effect of a network, which increases the value and attracts new users. Such templates were available long ago, but their use became more widespread with the development of the Internet.

Google may be considered to be an example of a multilateral platform. The Google value proposition is the provision of targeted advertisement (AdWords) in the Internet for one of the segments of the customers – the advertisers. The success of this model depends on the presence of a large number of advertisement recipients – the users. A value proposition for the users is the possibility of viewing texts in the Internet, which other users may publish free of charge. The platform combines the authors of texts in the Internet, users searching the resources and the advertisers.

Free.
The Free Model assures the use of the offer free of charge to at least one of the customer segments. Costs of non-paying customers are covered either by other customer segments or by other elements of the business model. The free offer is available under three business schemas (Osterwalder and Pigneur, 2010):

- Multilateral platforms – A scheme consists in existence of few customer segments, one of which obtains a free service or product. Subsidising costs of one of the customer segments are covered by another segment. An example in this respect is the free of charge paper called Metro, which is handed out to commuters in public transport means. Costs of free of charge distribution are covered by the advertisers.
- Freemium – This scheme assures a gratuitous basic service and an extended service at a charge. The gratuitous service is to attract the customers, and some of them will need the extended offer, which they will have to pay for. For example the Flickr portal allows access to a free of charge service of publishing pictures. It is used by occasional users. As regards users requiring additional options (such as for example bigger account capacity) the portal offers a Pro account at a charge, with an annual subscription. Costs of the platform, brand promotion and handling non-paying customers are covered by paying customers.
- Bait and hook – The basis of this scheme is to provide bait for the buyer. As a rule the bait comprises the possibility of cheap purchase of a product or service at a price below the production costs. Such purchase means taking the bait, and namely taking up a commitment to beating constant purchases by the customer. An example of such a scheme may be the Gillette razors. The starting set is the bait. After buying the razor, the consumer have to bear regular costs related to buying blades, the price of which is not as attractive as in the case of the bait.

Open business models.
Innovation is becoming an increasingly open process thanks to a growing division of labour. One company develops a novel idea but does not bring it to market. Instead,

the company decides to partner with or sell the idea to another party, which then commercializes it. To get the most out of this new system of innovation, companies must open their business models by actively searching for and exploiting outside ideas and by allowing unused internal technologies to flow to the outside, where other firms can unlock their latent economic potential (Long Tail, 2015).

Open business models are used to generate value by cooperation with external partners. An important factor spurring the process of open innovation is the rising cost of technology development in many industries. There's a second force at play - the shortening life cycles of new products. Two types of such cooperation are possible:

- From inside out - Companies attract the best specialists from the sector to develop R&D concepts and deploy solutions on the market. In the 'from inside out' model the company sells concepts which have not been used yet in the organisation.
- From outside to the inside - Companies employ good specialists, but they are aware that not everybody works for them. They perceive external R&D as a source of value. In the model 'from the outside to the inside', the company buys concepts from the outside. They are also of the opinion that internal R&D is necessary due to retaining a part of the value generated by external R&D for itself.

A good example of such activities is GSK, which sells concepts that had not been used to the outside. Another example is pharmaceutical drug development. Investment in a successful product has risen to well over $800 million, up more than tenfold from just a decade earlier. Even the consumer products industry is feeling the pressure. P&G estimates that its Always brand of feminine hygiene pads, which cost $10 million to develop a decade ago, would set the company back anywhere from $20 million to $50 million today (Long Tail, 2015).

To resume, it is worth to note some novelty of the research and obtained results. The material and cultural phenomena and processes within industrial networks were considered simultaneously. This enabled new modeling approaches, and much more relevant explanations of the discussed phenomena were obtained, than before. The results suggest new types of threats and perils, which may follow lean networking, global outsourcing and off-shoring. Finally, a new way of parallel researching of social phenomena observed in media and operational processes observed in the operational databases was proposed.

6 Assessment

The main objective of the assessment is to allow the selection of one of the models developed in the preceding step. When applying the approach propagated in the Blue Ocean Strategy it should be verified whether the created model has reduced costs and concurrently increased value for the buyers. As the value for purchasers arises from usefulness and price offered to the buyer by the company, and because the value for the company is an effect of its prices and costs, innovation in value is only achieved when the entire system of usefulness, prices and costs is appropriately planned in the company (Chan Kim and Mauborgne, 2010).

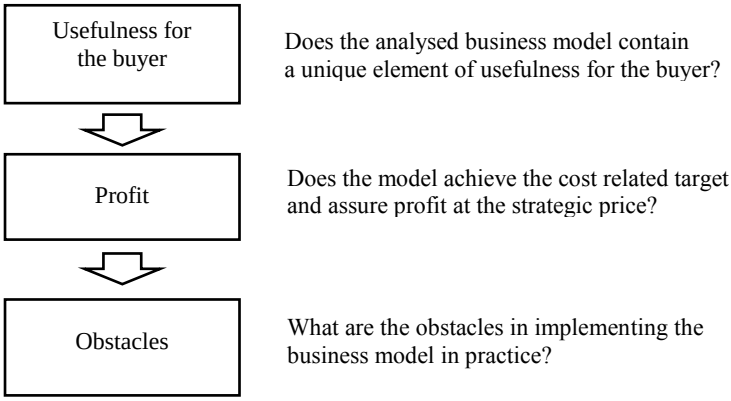


Fig. 4. Assessment sequence of a business model **(**based on Chan Kim and Mauborgne, 2010)

Fig. 4 presents the sequence of a business model assessment. In the first step an analysis of the usefulness for the buyer is carried out. It comprises granting an answer to the question whether the value proposition delivers a unique usefulness for the customer segments. In the next step it should be considered whether the offer is likely to assure the appropriate demand and attract customer segments that would generate sufficient revenues at the strategic price. An endeavour at securing profits brings the analysis to the cost. In this step it is also necessary to find a response to the question whether there is a possibility of achieving profit at the assumed costs. In the last step it is necessary to overcome obstacles in implementing the concept in practice.

Usefulness for the buyers.

The developed strategy backdrop presents strategic factors as compared to the sector. The competitiveness factors presented on the strategy backdrop should ensure exceptional usefulness for the customer segments. The mere visualisation allows for evaluation of the strategic profile. This is done with the use of the following:

- Concentration point - Does the strategy backdrop indicate only some competitiveness factors, because it is impossible to make outlays in all factors if the strategic price is maintained.
- Divergence - Does the designed strategy backdrop differ considerably from the average profiles in the sector. Only a significant difference allows the customers to notice the unique usefulness.
- Final message that rivets attention- Is it possible to formulate the value proposition presented on the strategy backdrop into a final message capable of drawing the desired attention. The final message can also be presented as a story, which is likely to convince the customer.

Should it turn out that the strategy backdrop has features of a good strategy, it is possible to assess its usefulness for the buyer, i.e. whether and in what way the product or service can change the buyers' lives. Table 2 presents the map of usefulness for the buyer, which should facilitate the assessment. Impressions of the buyers can be

Table 2. Map of usefulness for the buyer (based on Chan Kim and Mauborgne, 2010)

		Purchase	Delivery	Use	Additives	Service	Sale
At what stage are the biggest lock	Customer productivity						
	Simplicity						
	Convenience						
	Risk						
	Pleasure/image						
	Environment friendly						

divided into a cycle comprising six stages, starting from the purchase and ending with the product sale. Each stage supplies a whole assortment of impressions, which were presented on the vertical axis. Places with the biggest blockages of usefulness for customers and non-customers should be marked on the map, and then it is necessary to consider whether the new usefulness allows non-customers to become customers. Next the offer should be analysed from the viewpoint of eliminating the biggest obstacles for usefulness. Once those data have been applied on the map it becomes visible whether the offer differs from those of other players.

Profit.

There are many measures of profitability. As a group, these measures enable the analyst to evaluate the firm's profits with respect to a given level of sales, a certain level of assets, or the owners' investment. Without profits, a firm could not attract outside capital. Owners, creditors, and management pay close attention to boosting profits because of the great importance placed on earnings in the marketplace (Gitman, 2003).

It should be borne in mind that costs cannot shape the price. The starting point for the profit assessment should be the strategic price from which the target profit margin is deducted and in such a way the target cost is fixed. In the evaluation process of the strategic price attention should be drawn to the fact that price mechanisms depend on the positions of the seller and the buyer (Berthon and Hulbert, 1999). In order to achieve the cost objective which would assure profit for the company, three levels can be used:

- Cost-related stream-lining and innovations - The methods comprise replacement of raw materials required for the production by less expensive ones, elimination of costly actions with a low added value, and transfer of the manufacturing process to cheaper locations.

- Partnership - The available activities comprise take-overs to al-low ensuring access to competencies, quick and effective assurance of abilities gained to the effect of scale.
- Price innovation - Price maximization is the most important goal for most corporations (Brigham and Houston, 2013). Should it prove to be impossible to achieve the profit by applying the two above described levers, other price mechanisms shall be researched/analysed, e.g. basing the model on lending, renting, system of temporary sharing or share in revenues.

If the company's offer (value proposition) is capable of assuring effective implementation of the profit side of the business model, it means it is actually ready for implementation.

7 Obstacles

Even the best project constitutes a change for the enterprise and may consequently cause certain concerns and resistance among the three groups of stakeholders: employees, partners and the business environment. By the time the company spends money on a new investment, it must first overcome such concerns. Awareness of their existence is the key to overcoming them. Consequently in this particular step it is necessary to consider the potential hazards caused by the stakeholders, possible reactions and their consequences for the business model being deployed in practice.

Osterwalder proposes to apply the SWOT analysis to evaluate the business model. The adoption of this method allows the execution of a factual evaluation and assessment both of the business model as a whole, as well as of its particular elements. While performing the analysis, important questions should be posed. What are the strengths and the weaknesses of the organisation. The responses should make us consider what the internal state of the organisation is. The remaining questions concern opportunities of the organisation and the hazards it has to cope with. What is more, the responses should also induce to make an assessment of the company in its external environment. Such an analysis would be a starting point for further discussions.

A negative result of the assessment suggests that there was mismatching of prices, costs and the offered value proposition. To allow a correction of the business model, one should go back to the analysis stage and consider the following:

- reconstruction of the value proposition
- reconstruction of market boundaries
- analysis of non-customers, i.e. new segments which may be attracted by the new value proposition
- comprehension of the customer to allow the creation of a better value proposition.

8 Planning the subsequent steps

Once the final model has been worked out, the implementation phase takes place. This is connected with identification of all activities, which would support the deployment of the model, as well as projects under implementation that may be in conflict with the model being deployed, and also the establishment of the appropriate legal structure. Furthermore, plans should also allow for quick development of business and actions that have to be performed if the business starts to develop quickly.

Monitoring of the deployment of a new business model and the possibility of effective introduction of relevant modifications are particularly important processes.

As regards contacts with stakeholders, the way of talking about the business model being deployed is of importance. A popular form used for this type of tasks is telling a story. People tend to respond much better to stories than to logical arguments. The story should be simple and have a single hero. The said hero should present the viewpoint of the customer or the company. Presenting the way in which the company has managed to solve the customer's problem is a key point in the story.

Below there is an example of a story scenario:

- Who is the example customer?
- What problem or need does he/she have?
- What solution did he/she use to date and what did he/she miss?
- In what way did he/she find out about the offered solution?
- How did he/she make the purchase?
- What is the basis of the solution?
- What benefits has the customer gained?
- What do the employees and acquaintances say about the product or the service?

9 Conclusions

As a result of studying available literature and on the basis of author's experience in designing business models the process of business model development was proposed. The main objective of executing the entire process is the selection of one prototype from all the developed ones. The selection consists in verifying whether the developed model reduced costs and at the same time enhanced value for the buyers. As for the buyer the value arises from usefulness and the price offered to the buyer by the company, and as the value for the company is an effect of its prices and costs, innovation is achieved in value only when the entire system of usefulness, prices and costs is appropriately planned out in the company. Consequently the developed business models are different from those adopted in the sector to date. These are models that implement real needs of the customers, and allow placing an innovative product on the market. Familiarity with the development of models and their assessments would also allow making investments in companies, which are operating based on the most competitive business models, indicating strategic differences that may determine gaining a competitive edge.

Particular areas require further development. This concerns the working out of scenarios and methods for their analysis, methods of analysing the business model environment, and development of business models from the viewpoint of analyses of innovation epicentres.

References

Anderson, C. (2006). The Long Tail: Why the Future of Business is Selling Less of More, New York: Hyperion.

Berthon, J.-P., & Hulbert J. (1999). Brand Management Prognostications, *Sloan Management Review*, Vol. 40: 53-55.

Brigham, E.F., & Houston, J.F. (2013). Fundamentals of financial management. Cengage Learning: 16.

Chan Kim, W., Mauborgne, R. (2005). Blue Ocean Strategy. Harvard Business Press: 36-50, 185-191.

Chesbrough, H.W. (2007). Why Companies Should Have Open Business Models. http://sloanreview.mit.edu/article/why-companies-should-have-open-business-models/. Accessed 2015.01.15.

Christensen, C.M., & Raynor, M. E. (2008). Innovation and the general manager. Harvard Business Press: 49-73.

De Bes, F.T., & Kotler, P. (2013). Winning at Innovation. The A-to-F Model. Rebis: 8-26.

Gitman, L.J. (2003). Principles of Managerial Finance. Addison Wesley, p. 62.

Lindgardt, Z., Reeves, M., Stalk, G., Deimler, M.S. (2009). Business Model Innovation. The Boston Consulting Group, p. 2.

Long Tail (n.d.). http://en.wikipedia.org/wiki/Long_tail#cite_note-8. Accessed 2015.02.15.

Osterwalder, A. (2004). The business model ontology. A proposition in design science approach. PhD Thesis, University de Lausanne, p. 14.

Osterwalder, A. (2013). Business Model – The Empathy Map. http://escuelaformacionupm.files.wordpress.com/2013/02/empathy-map-poster.pdf. Accessed 2014.IX.11.

Osterwalder, A. (2014). Value Proposition-Design. http://www.businessmodelgeneration.com. Accessed 2014.IX.01.

Osterwalder, A., & Pigneur Y. (2010). Business Model Generation, John Wiley & Sons: 18-31.

Quinion, M. (2011). Turns of Phrase: Long Tail, http://www.worldwidewords.org/turnsofphrase/tp-lon1.htm. Accessed 2014.XII.30.

Sierpińska M., Jachna T. (2004). Rating companies using global standards (in Polish). PWN: 50, 286-287.

Teece, D. J. (2010). Business Models, Business Strategy and Innovation. *Long Range Planning*, Vol. 43: 172-194.

Business Model Visualisation for ICT Product

Robert Wojtachnik

Warsaw University of Technology, Warsaw, Poland

wojtaro@wp.pl

Abstract. Taking into account the multi-dimensional nature of areas to be analysed, business consultants sought simple methods that would allow the visualization of a business model. This chapter presents visual business models presently used by consulting companies. The three most popular models discussed herein are: BCG, Business Model Canvas and Lean Canvas. As business models are designated for visualisation of already existing ICT products, the further part of the chapter outlines the Business Model Canvas. It comprises nine elements divided into four elementary areas: the offer, the customers, the infrastructure and the finance. Execution of all steps related to the creation of the business model proposed in this chapter will guarantee that all important modelling aspects would be taken into consideration.

1 Introduction

For 60 years companies have been seeking a way to establish value for customers and to assure that profits would be gained from a value established in this way. The search for the above mentioned business method arose from problems faced by boards, and the development of a model that could be accepted by the market determined whether the enterprise could survive. The story of the concept called the business model is inextricably linked with the history of innovations, the commercialisation of which was not easy due to the timeless and breakthrough nature of their use, so divergent from the current market circumstances. The history of mankind is also the history of inventions, for which application had to be found. The light bulb invented by Thomas Edison would not have changed history, if its inventor had not discovered for it the entire market surroundings and ways of its application.

In 1959 Xerox manufactured the first photocopier, which was to replace manual copying of texts. Yet it proved to be too expensive for it to be placed on the market. A consulting company was then hired to develop a commercialization plan for the invention. The consultants reached the very same conclusions as Xerox and recommended the withdrawal of the copier from production. Xerox was sure of its invention and started to seek a model suitable to allow it to place the device on the market. The concept assumed that the device would be leased, with a certain set limit of copies under a leasing instalment price. Once the copy limit has been exceeded, the lessee would have to pay for each copy made. The new concept was accepted by the market,

and the users made thousands of copies. Selling dozens of thousands of its devices, Xerox joined the largest companies worldwide (Osterwalder, 2014b).

Other problems in the commercialization of their concepts were also encountered by Internet companies. By 1998 the browser presented search results on the basis of the frequency of appearance of the given word or phrase on a page. In 1998 Google made its debut in the Internet along with its browser. The core of the search system was PageRank, which referred to the network of mutual links, and which assessed each page.

In 1998 Microsoft bought LinkExchange, a company specialised in the distribution of advertisements between particular pages. The company purchased a license from Scott Banister for the concept of a programme, which set up auctions for phrases in the browsers. In 2000 Microsoft employees, following Banister's concept, initiated an experimental system that matched advertisements to the phrases sought. In the same year the project was closed for fear of loss of revenues of Microsoft from large-format advertising. In 2002 Google presented the AdWords programme, based on Banister's concepts. It turned out that a similar product has already been patented by a company called Overture, which was later taken over by Yahoo. In 2004 Google obtained from Yahoo a license without a time limit for the patent in exchange for 1% of the shares.

In 2003 Microsoft started works on a new browser and an advertisement system based on search results, which was marketed in 2004 under the name of Moonshot. At that time the Google project was already much more advanced. As a result the project of Yahoo and Microsoft was left far behind Google, and the considerable part of advertisers recognised the advantages of this system. In 2008 Google's revenues constituted 75% of the sum generated by advertisements that accompany search results, while Yahoo achieved 20% and Microsoft 4% (Brandt, 2011).

According to Marketing Week (Marketing Week, 2009), 44% of business leaders admit that they are familiar with basic tools for the incorporation of innovations in organisations. As regards innovations, companies cannot resort to the commonly recognised management model. The situation is different in other areas, which have been distributed between clearly defined units and staff, which have at their disposal well known and appropriately developed methodologies and tools (De Bes and Kotler, 2011). This chapter endeavours to describe a method of establishing a business model used for the implementation of the already existing ICT technologies.

2 Review of business models

Development in the era of globalisation led to changes in the traditional balance between the customer and the vendor. New ways of communication, and the development of the Internet technology led to a situation, in which the customers have at their disposal a much bigger selection of products and they may much more easily compare their features. This is why enterprises should strive to be oriented at the customer (customer centricity). This development direction requires the enterprises to redefine the value offered to the customer and withdrawing from the hitherto used business model (Teece, 2010).

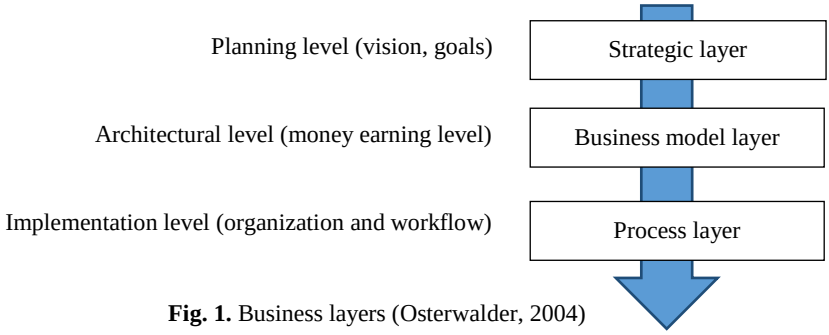


Fig. 1. Business layers (Osterwalder, 2004)

The concept of the business model was initiated by innovations in the IT sector, and basically this model was meant to be less formal than the strategy formulated as a strategic plan (Koźmiński, 2004). Alexander Osterwalder suggested that the business model describes premises for the way in which an organisation creates value and assures and gains benefits from a value established in such a way. The business model consists in an outline of strategy, which is to be incorporated under the structures, processes and systems in a given organisation (Osterwalder and Pigneur, 2010). Fig. 1 presents the position of a business model within an organisation. The exhibit shows that the business model layer is between the strategic and process layers, and also explains the way in which an enterprise is able to earn money. As an effect the developed business model should help the company to find an answer to the question: How do we intend to win in this business (Welch and Welch, 2005)? Jack Welch's experience (as the CEO of General Electric) shows that this question may be answered by formulating a mission of the organisation.

One of the first contemporary visual formalised methods of describing business models was the proposal of BCG (The Boston Consulting Group) shown in Fig. 2. According to it the business model consists of two elements – the value proposition and the operating model. The value proposition provides an answer to the question what is offered by the company and to whom, in three dimensions respectively:

1. Target segments – which customers would the value proposition be offered to and which needs would it satisfy,

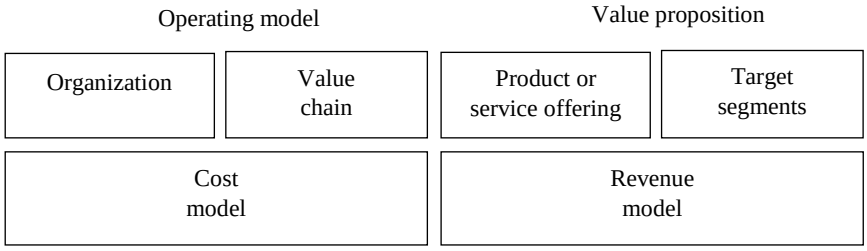


Fig. 2. A business model consisting of six components **(**based on: Lindgardt and Reeves, 2009; Stalk and Deimler, 2009)

2. Offered product or service – what is offered to the customers to allow satisfying their needs,
3. Income model – ways of gaining profits from the offer value.

The operating model provides an answer to the question what is the profit at which the company provides the offer. BCG delimited three areas influencing the result achieved by the company:

1. Value chain – describes the way in which the company responds to demand shown by the customers, and also which processes it intends to handle by it-self, and which it plans to assign to external companies
2. Cost model – describes the way in which assets are to be organised and costs are to be planned to be able to provide a value proposition at a profit
3. Organisation – ways to organise the work of the staff to maintain or enhance the competitive edge of the enterprise.

In 2010 Alexander Osterwalder and Yves Pigneur published a book entitled "Development of business models". Preparation of the book engaged a group of 470 managers from 45 countries, who were members of the Business Model Innovation Hub. They presented their own cases for needs of the analysis, and also assessed and commented the first versions of the book. As a result a practical guide was developed for people who are considering designing enterprises of the future or changing outdated business models. This concept was applied in such companies as IBM, Ericsson, Deloitte, NASA, GE.

The basic tool used for visualisation of a business model is the Business Model Canvas, which was presented in Fig. 3. It comprises nine elements divided into four basic areas: offer, customers, infrastructure and finance. These elements include:

1. Customer Segments - groups of customers which an organisation serves or intends to serve
2. Value Proposition - solving problems or satisfying the needs of customers with the use of the value proposition
3. Customer Relationship - represent the type of relationship taken up with particular customer segments
4. Channels - channels (of communication, distribution or sale) via which customers are provided with the value proposition
5. Revenue Streams - effect of delivering the value proposition to the customer segment

<table>
<tr><td rowspan="2">Key Partners</td><td>Key Activities</td><td rowspan="2">Value Proposition</td><td>Customer Relationship</td><td rowspan="2">Customer Segments</td></tr>
<tr><td>Key Resources</td><td>Channels</td></tr>
<tr><td colspan="3">Cost Structure</td><td colspan="2">Revenue Streams</td></tr>
</table>

Fig. 3. Business Model Canvas (based on Osterwalder and Pigneur, 2010)

6. Key Resources - assets necessary to generate the value proposition, its delivery to the customer segments via relevant channels and maintaining relationships with customers
7. Key Activities - refer to works which have to be performed for the model to function as required
8. Key Partners - conform to resources obtained from out-side the enterprises and actions executed outside
9. Cost Structure - costs connected with the development and delivery of the value proposition

An alternative model for the above presented one is the Lean Canvas developed by Ash Maurya, which negates the excessive simplicity of the Osterwalder's model. The development of the Lean Canvas model as based on a review of actions executed by companies which had been successful before they managed to reach the very top. This model is more focused on problems and risks of a specific company concerning introduction of a new product. The second tier on which this model has been based on is the presentation of concept in the product-market system. This model also consists of nine elements (Table 1) and is presented in Fig. 4.

Table 1. Elements of Lean Canvas

Problems	definition of three problems of potential customers (customer segments)
Solution	specification of three key properties of a product/service that allow solving the defined problems
Key Metrics	key actions, which need to be monitored
Cost Structure	costs connected with the manufacturing and maintaining a product, gaining customers and providing a value proposition
Customer Segments	groups of customers which an organisation serves or intends to serve
Unique Value Proposition	individual and clearly specified proposition in the form of a short message explaining how the value proposition differs from that of the competition and why it is worth the requested payment
Competitive edge	specifies whether a given product/service can be easily copied or purchased on the market
Channels	channels (of communication, distribution or sale) via which customers are provided with the value proposition
Revenue Streams	effect of delivering the value proposition to the customer segment

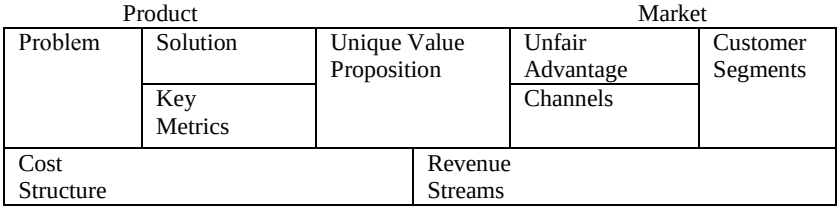


Fig. 4. Lean Canvas (based on: Maurya 2012; Maurya 2014)

According to Maurya (2012) a lot of companies are unable to reach success, because they operate in conditions of extreme uncertainty and dedicate their time to the development of faulty products. Success requires comprehending a maximum of three problems, to which we seek answers by developing a product and identify the relevant needed solutions (Pucher 2012, Maurya 2012).

The following part of the chapter will outline ways in which formulated elements of the business model template can be defined.

3 Method for business model description

In order to define the business model, the Business Model Canvas used, the objective of which is to visualise the key nine elements. The described model is not the only feasible and final project. It should be considered as a description of the existing state or a concept of the way in which a product could be commercialised.

Customer segments.
The customers constitute the basis of existence of each enterprise. To be able to better satisfy needs of the customers, the enterprise may group them into segments by common characteristic traits, similar forms of behaviour or properties. A group of customers may be considered to be a separate segment when:

1. Needs of a group of customers require formulating a specific type of offer,
2. They require access via separate distribution channels
3. It is necessary to establish a different type of relation
4. The group stands out with respect to the profitability structure

One or several customer groups are to be defined within one model, and then it is to be indicated which market segments are to be served (so-called targeting) and which are not to be. When identifying a set of segments, one of one of two options may be used: focus on one segment or a decision to serve a large number of such segments, addressing a different offer at each one of them. Making such a selection will allow working out a business model which would meet the needs of a specific customer segment. Designers of a business model should take into consideration who is the value generated for and who the key customers are.

Below possible ways of distinguishing particular customer segments are listed:

1. Mass market – No segmentation is done for handling the mass market. The offer is focused on one large group of recipients characterised by similar problems and features. An example in this respect is an offer of electronic equipment for all natural per-sons.
2. Niche market – In this model companies decide to serve a specific customer segment. A good example in this respect is an offer of exclusive wines addressed at presidents, CEOs presented during wine tastings organised in those enterprises.
3. Segmentation – A division into customer segments characterised by different needs and problems. An example may be the segmentation made by banks which

divide customers by their income. A different offer is addressed by the banks at each of those segments.

4. Diversification – An organisation operating according to this model serves at least two customer segments and offers to each of them an individual value proposition. Diversification is also a way of reducing risk in business activity. The Amazon trading company had considerable IT resources at its disposal for needs of serving the biggest Internet shop worldwide, which it decided to use by offering the cloud computing service.
5. Multilateral platform (multilateral market) - This concept is based on serving two customer segments, which are mutually needed to allow the company to operate in an effective way. An example of this model is the distribution of a gratuitous paper to the traveller segment, which contains advertisements of the second customer segment – and namely the advertisers.

Value proposition.

The value proposition is the reason why customers decide to choose an offer of one company over others. A characteristic feature of a value proposition is solving problems of a given customer segment or satisfying their needs. It describes the set of products and services that generate value (benefits) for a given customer segment. Elements that generate value comprise those of a quantitative and a qualitative nature (Osterwalder and Pigneur, 2010):

1. Novelty/innovation – Satisfying a new set of needs by a new value proposition. One of the examples may be the introduction of a mobile telephone, which enhances availability in the field.
2. Output – This is one of the traditional ways of generating value. It is based on increasing the product output. One of the examples is increasing the performance of computers.
3. Tailoring to individual needs – The cause for generating value for the given customer segment is in this case the adaptation of a product to concrete needs of that segment. An example in this respect is the implementation of ERP systems, which allow their adaptation to specific processes adopted in the customer's organisation.
4. Effectiveness – The key to success is supporting the customer segment in the execution of their specific tasks. A good example in this respect is the vehicle route optimisation system, which enhances the effectiveness of the employees by speeding up the execution of their tasks and reducing the ensuing costs.
5. Pattern designing – A value for a group of customers is the unique design of the products, which is a frequent value proposition on the automotive or the clothing markets.
6. Brand and status – A source of value for the customer is the fact of using a specific brand, which increases its user's status and emphasises that its user belongs to a given group. An example of such brands is Rolex.
7. Price –This is one of the ways in which segments of price sensitive markets may be satisfied. It consists in delivering value at a lower price. It is necessary to be

fully aware of the consequences ensuing from such a value proposal, which will affect the remaining elements of a business model. A good example is the development of an offer of cheap flights by Ryanair.

8. Low costs – Allows the customers to decrease their operating costs. A good example in this respect in the offer of programmes under the SaaS (System as a Service) model, the objective of which is reducing costs of application deployment.
9. Lower risk – Consists in offering a value proposition, which would enable the customers to reduce the risk connected with purchasing a product or a service. For example, granting an annual guarantee for used vehicles reduces such risk.
10. Availability – A concept for the value proposition is offering accessibility to a product or service for a group of customers, who earlier on had no such possibility. Expanding the possibilities may be connected with a change of the business model or the appearance of new technologies. For example, NetJet offers a partial ownership of jet airplanes.
11. Convenience and usefulness – Value for the customer may also depend on better convenience of usage. An example of such value was the introduction of the iPod and launching iTunes which made the process of searching for and purchasing music items much easier.
12. Experience – A value may also be constituted by experience of a company in the implementation of specific services or products. This is one of the tiers in the operation of consulting companies.
13. Design – Emphasising product qualities by its unique design, which may be implemented for example in the sector of electronic appliances.

The value proposal may be visualised by applying the strategy backdrop, which allows pinpointing the current situation (competition factors) in the sector. It allows comprehending which competition factors are invested in the competition, and what do the customers gain under a competitive offer on the market. The strategy backdrop may be presented in a graphical form. On the horizontal axis factors related to competition and investments in the sector are to be provided. The above presented list of factors that generate value for the customer may be used to identify those factors. The vertical axis of the strategy backdrop depicts the level of the offer oriented at buyers in relation to each of those key competitiveness factors. A high price means that a company offers more to the buyer, and consequently makes bigger investments in the given factor. Points conforming to the offer of particular competitive companies should be applied on the diagram. It is also possible to develop a strategy backdrop for the average in the sector (Chan Kim and Mauborgne, 2005).

4 Customer channels

Customer channel shows the way in which a company communicates with particular segments of its customers and provides them with a value proposition. The type of relation that is entered by a company with its customers depends on the determination of communication channels. The channels indicate points of contact of a customer with the company and affect the impression it gets. Those channels allow the custom-

ers to purchase a given product or service and enable executing value proposals for the customer. A company may contact the customer via own channels (direct or indirect ones) or via partner's channels (indirect ones). Direct channels mean sale through a corporate shop and Internet sale. Indirect channels comprise own shops, partner's shops and wholesalers. Fig. 5 shows the functioning phases of a channel, which start with making the customer aware of the existence of the given products or services.

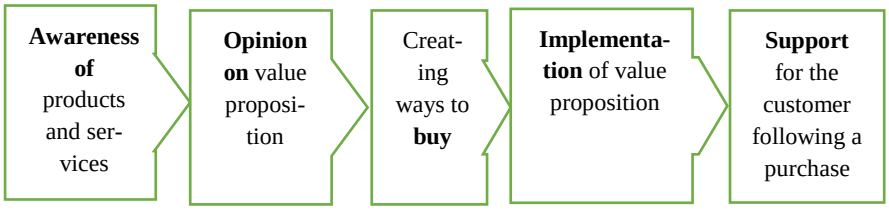


Fig. 5. Functioning phases of a channel **(**based on: Osterwalder and Pigneur, 2010)

Then the customer forms an opinion concerning the value proposition. Once the customer has been encouraged to buy the product/service, the purchase possibility has to be established and a value proposition provided. The last element in the functioning of a channel is support for the customer in the usage period. Also the below questions may be used in the process of defining the channels (Osterwalder and Pigneur, 2010):

- Which channels should be used when establishing contact with particular customer segments?
- In what way does the company establish contacts?
- Are the channels integrated?
- Which channels turn out to be the best?
- How do particular channels conform to the standard rules of conduct with the customers?
- Which are the most economic ones?

5 Relations with the customers

The basic target of an enterprise is to gain customers, keep them, and generate a sales increase. This is made possible by determining the type of relations it would like to establish in particular segments of its customers. The relationships may be extremely divergent – from fully automated ones (such as platforms of Google AdWords) to completely personal ones (e.g. insurance agents). Also of importance here are expectations of the customers, costs required to build and maintain the relations as well as the integration level of actions of relation building with other elements of the business model.

An enterprise may build different types of relations with the customers, and may participate in one of them or in several:

1. Dedicated customer assistant - In this concept one employee is assigned to provide assistance to a customer. This type of relation requires certain time for its establishing, and is the closest form of a commercial relation. A good example may be a dedicated assistant to bank customers who are holders of a golden credit card.
2. Personal assistance - This particular type of relation is based on human interactions. The assistant supports the customers in the purchase process, as well as after it. A good example of this relation is support provided in the purchase of various types of tariff plans, such as for the television and in telephony.
3. Self-service - The company does not establish direct relations with the customer. All it does is provide the customer with the necessary tools to allow individual usage of the offer. A good example may be small self-service retailers where the customers are offered carts and self-service cashier points.
4. Automated service - This service combines elements of self-service with automated services. It allows the identification of a specific customer and tailoring the offer to its preferences. One of the examples may be automatic recommendations of titles of books/DVDs/films appropriate to the preferences of a given customer.
5. Communities - Making use of the potential offered by the community to forge closer contacts with the customers. Such mechanisms allow the community to share the knowledge and provide mutual assistance in problem solving. They are also used by companies to gain knowledge about specific problems of communities to be able to manage expectations in a more effective way.
6. Joint development - This concept assumes building value together with the customer. Such a relation may consist in encouraging clients to post opinions, share contents (YouTube.com) or – as in the case of Lego – establishment of a portal through which the customers may create individual sets of blocks, and the best ideas would be adopted in mass production.

6 Revenue streams

According to a business model being developed, the company serves one or more customer segments by providing them with a defined value proposition. The customers are willing to pay for obtaining this value. The company generates revenue streams in connection with serving each of the customer segments. Two mechanisms of revenue stream generation may be determined. The first one of them assumes gaining transaction streams arising from the execution of individual transactions. The second one is based on periodical constant revenues resulting from executing value proposals or from providing post-sale support. Each of the generated streams may be based on a different price mechanism. Below the possible ways of generating streams are listed:

1. Sale of assets – This method consists in the sale of all proper-ty assets comprised by the balance sheet in value terms (Sierpińska and Jachna, 2004). The sale of assets should be understood as the sale of the ownership title to a specified product or goods, which is included in the balance sheet in current assets. One of the examples is the sales of vehicles or vehicle parts.

2. Fee for usage – This is a fee charged to the client for using the service. The value of the fee depends on the frequency or the used quantity. For example, telephony operators charge a fee for the time of calls made.
3. Subscriber fee – This method consists in payment in return for constant access to a specified service. For example, Internet operators charge a fixed monthly fee for access to the Internet.
4. Rent and leasing – The revenue stream appears when the customer obtains an exclusive right to use the de-fined resource within a specified period of time for a fixed fee. The party that uses the given commodity does not have to bear full costs of having the given commodity, but merely periodical costs of its usage. This is the basis for the operation of companies which rent or lease vehicles.
5. Licence granting – This concept is based on allowing the possibility of using a commodity against a licence fee. The licence does not require the commodity to be produced again and assures to the author/owner the right to sell it several times. This type of fees is quite popular in the media sec-tor or in the technology sector (access to patented technology).
6. Intermediation commission – When one of the parties provides an intermediation service to other parties, a commission may be charged depending on the quantity or value of the intermediation object. This applies for example to collecting a charge for credit card payment as interest on the transaction value.
7. Advertisement – The revenue stream is generated by publishing an advertisement of a given service/product/brand. The advertisement is a popular source of revenues in the media sector. The customers pay either for broadcasting of the advertisement (with the price depending on the duration of the emission and its time) or the effects (the YouTube portal collects a charge only per viewed advertisement).

According to Berthon and Hulbert (1999) the price mechanisms depend on the position of the seller and the buyer. Fig. 6 presents the possible price mechanisms dependent on the described possibilities.

Chan Kim and Mauborgne proposed a tool for the valuation of products/service, which they called a mass price corridor (Chan Kim and Mauborgne, 2005). The fixing of a price requires going through the following three steps for each of the proposed streams of revenues and the strategic group:

1. Seeking three types of alternative products or services. The first one of them comprises products of the same form, which come from the analysed sector. The second type includes services of a differing form, but the same function (products of a similar utility but a divergent physical form). In the third group there are products with a different form and functions, but intended to serve the same objective.
2. Illustration of prices and volume of alternative products and services. In this step circles with a differing radius are to be drawn. The radius of a circle illustrates the

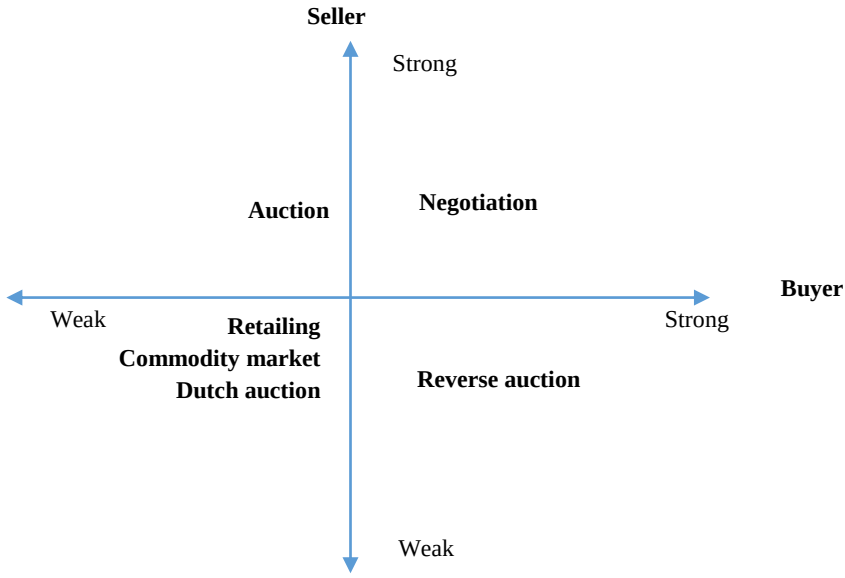


Fig. 6. Price strategies (based on: Berthon and Hulbert, 1999)

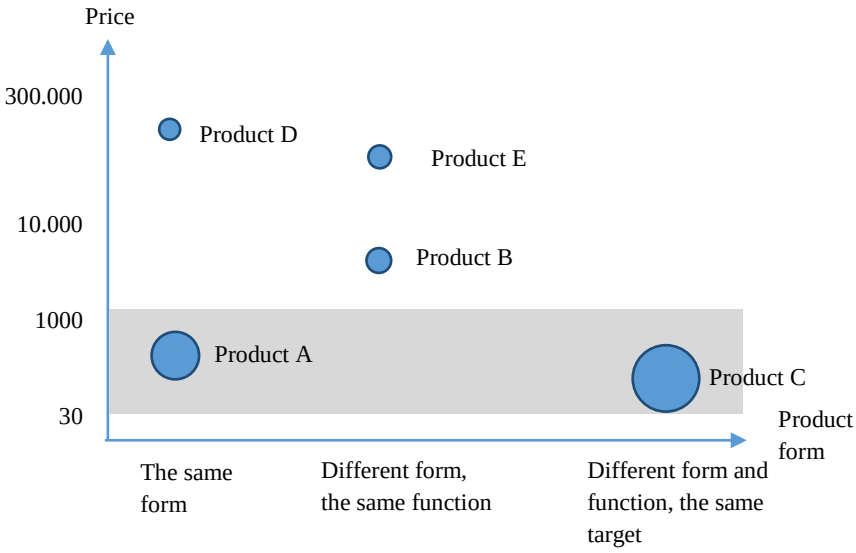


Fig. 7. Visualisation of a mass price corridor (based on: Chan Kim and Mauborgne, 2005)

number of buyers of a given product or service. The last step is the drawing of a mass price corridor understood as the biggest group of target customers. A sample mass price corridor is presented in Fig. 7.

3. Fixing the price level inside the corridor. The process takes into consideration all factors which make imitation impossible, i.e. the coverage of a product or service by copyrights/patents and exclusivity for the possession of an asset element or relevant skill.

If the actions proposed in the above item are performed, revenue streams from each customer segment may be determined as well as the strategic price for each of the revenue streams. The fixed strategic price is a benchmark in the determination of the cost level.

7 Key resources

A company has to involve certain resources to be able to deliver proposed values for the selected customer segments. Plans are to be drawn up as to the resources required also by distribution channels, relations with the customers and revenue streams. In this element of the business model the most important resources have to be indicated that are necessary to assure its correct functioning, and which may be divided into the following four groups:

1. Physical resources - They comprise such assets, as production infrastructure, buildings, vehicles, sale network, equipment and IT systems.
2. Intellectual resources - In the opinion of numerous companies the brand is their main resource. It is considered the main asset for clothing companies, shoe making companies and luxury consumer goods manufacturers. The capital of the brand may be considered as a sort of a storage area of future profits of the company, connected with earlier investments in the brand (Panfil and Szablewski, 2009). In the technological sector of the greatest income from the license. As regards trade companies, their most valuable intellectual resources consist in the customer base and alliances with the partners.
3. Human resources - The key resources in sectors that require specialised knowledge (e.g. in the pharmaceutical sector) or special skills (e.g. sale skills).
4. Financial resources - Some business models require making considerable outlays, the function of which is to secure the risk of the creditors (Panfil and Szablewski, 2009). Taking into consideration the origin of the capital, its sources may be divided into own source and external sources (i.e. credit). Own capital may come from contributions made by owners (cash) or from retained profits. Also external capital may be used to finance the development of an enterprise, the source of which may comprise a bank credit, emission of ordinary bonds and convertible debentures, the issue of securities, leasing, and merchant credit from the suppliers (Sierpińska and Jachna, 2004).

The matrix presented in Table 2 may be useful along the process of defining resources.

Table 2. Allocation of resources to elements of the business model

Resources	Value proposition	Distribution channels	Customer relationship	Income flow
Physical				
Intellectual				
People				
Finance				

8 Key activities

This element of the business model describes the key activities required by the value proposal. It is also necessary to plan the activities which may be needed by the distribution channels, relations with the customers and streams of revenues. In this element of the business model the most important activities need to be indicated which are indispensable to assure its correct functioning. They may be divided into three groups:

1. Production – These activities consist in designing, manufacturing, quality assurance and delivery of required amounts of the product.
2. Problem solving – Solving the problems of customers may be the basic operation of an enterprise (e.g. consulting companies) or processes that support the basic operation (e.g. assuring product servicing). Problem solving is inextricably connected with the processes of knowledge management and organisation of training courses for the staff, which should be taken into account in the cost structure.
3. Platform or network – In business models organised around a platform or a network the key actions comprise the development, management, assuring the continuity of the service and promotion of the platform. The platform function may be fulfilled by external resources (networks, partner dating services or programmes). Allegro is one of the platforms that associate buyers and sellers.

9 Key partners

When planning its business, a company may choose to implement its tasks on its own or may entrust their execution to external resources. Causes for the delegation of tasks to external companies and commencing a partnership may comprise the following:

1. Optimising and economies of scale – this is connected with cost reduction by joint usage of resources (e.g. outsourcing services, common server infrastructure).
2. Lowering the risk and uncertainty level – despite the fact that companies compete with each other in a single area, they may create alliances in other areas to conduct activity in a competitive environment (e.g. abroad) in conditions of lack of certainty. The objective of such an alliance is to share risk.
3. Adoption of specific resources or actions – only some companies may prove to have all the required resources which jointly form the business model. The remaining companies transfer the encumbrance ensuing from maintaining specific

resources onto other entities. This applies for example to the purchase of an operating system by computer manufacturers instead of developing their own.

However, not all partners can be considered to be those of key importance. The decision about recognising a partner as a key one depends on whether such a partners allows obtaining key resources or involvement into key actions. When taking a decision about a partnership, four types of relations may be entered. The first one of them is building a strategic alliance between companies which are not competitors. Competitors may establish a strategic partnership consisting in mutual competing (e.g. joint involvement in a tender). Companies may also jointly form new business entities, such as for example the opening of joint sale points by companies that manufacture complementary goods (e.g. telephone, the TV, the Internet). Companies may also enter into a partnership to assure the security of supplies (cooperation between the buyer and the supplier, which guarantees on the one hand secure deliveries at fixed prices, and on the other hand – the sale of products).

10 Cost structure

Major decisions with regard to operating cost structure must therefore focus on their impact on the firm's value (Gitman, 2003). This structure comprises all expenditures borne in connection with the usage of the business model described in the above items. First of all the most important costs have to be specified, which are generated by the business model. These costs may be divided into four main groups:

1. Fixed costs - costs which do not depend on the production volume, and their level is constant within a short or average period. Business risk depends in part on the extent to which a firm builds fixed costs into its operations - if fixed costs are high, even a small decline in sales can lead to a large decline in ROE (return on equity) So, other things held constant, the higher a firm's fixed costs, the greater its business risk. Higher fixed costs are generally associated with more highly automated, capital intensive firms and industries (Brigham and Houston, 2013). An example in this respect is rents, maintaining establishments, depreciation, leasing costs, fixed rate remuneration (Rutkowski, 2007);
2. Variable costs - these costs vary depending on the production volume. Here the firm would not have much automated equipment, so its depreciation, maintenance, property taxes, and so on would be low. However, the total operating costs line has a relatively steep slope, indicating that variable costs per unit are higher than they would be if the firm used more operating leverage (Brigham and Houston, 2013). An example in this case may be costs of direct materials, piecework remuneration, technological energy. Variable costs tend to grow with the production increase (Rutkowski, 2007);
3. Economies of scale - benefits which become available with the growth of production volume as an effect of a decrease in the cost of one-off manufacturing as the production volume grows. Economies of scale are enjoyed when the average unit cost of production goes down as production in-creases. One way to achieve econ-

omies of scale is to spread fixed costs over a larger volume of production (Myers et al., 2007). An example of such a factor are low prices of wholesale purchases, which owing to a large volume are lower than in the case of retail purchases;
4. Economies of scope - this type of benefits appears at the time when the scope of the operation is expanded as an effect of making some activities joint (e.g. marketing, financial and distribution channel related activities).

The cost structure should be considered from the viewpoint of strategic assessment defined in advance, which should be a benchmark for the determination of the cost level. As regards cost management, two types of structures may be determined – those oriented on costs and on values. As regards orientation on costs, key activities comprise cost reduction in all possible situations to assure positive profitability of the business operation on the assumption of revenues arising from the fixed strategic price. The process of cost reduction a review shall be made as to which resources and actions are most expensive, and afterwards the reduction process should be commenced from items with the highest costs. If we consider orientation on values, when cost reduction is impossible below a certain level, the only factor that could substantiate higher costs, and consequently price increase, may be a change in the proposed value (enhancement of its attractiveness).

11 Conclusions

The presented development method of business models was made possible thanks to the theoretical and practical studies of 470 managers. As a result of those studies a method was proposed for the development of business models, which consists in nine elements: customer segments, value proposition, relations with the customers, channels, revenue streams, key resources, key activities, key partners, and the cost structure. Contrary to the propositions of Osterwalder and Pigneur, the Business Model Canvas is only the backbone in which streamlining was applied with respect to creating value for the users, including also the Blue Ocean Strategy for visualisation of a mass price corridor. It was proposed to make use of the strategy backdrop to allow the visualisation of the value proposition, the process of establishing a price strategy was supported by the model of Berthon and Pitt, and a selection was made of the strategic price based on the mass corridor. In this chapter the method of allocating resources to elements of the business model was also described.

Particular areas require further development. They include: ways of combining the business model with method of analysing the business model surroundings, as well as further studies on the planning of distribution channels and value propositions.

References

Berthon, J.-P., & Hulbert J. (1999). Brand Management Prognostications, *Sloan Management Review*, Vol. 40: 53-55.

Brigham, E.F., & Houston, J.F. (2013). Fundamentals of financial management. Cengage Learning, 16.

Chan Kim, W., Mauborgne, R. (2005). Blue Ocean Strategy. Harvard Business Press, 36-50, 185-191.

De Bes, F.T., & Kotler, P. (2013). Winning at Innovation. The A-to-F Model. Rebis, 8-26.

Gitman, L.J. (2003). Principles of Managerial Finance. Addison Wesley, 62.

Kotler, P. (2014). Kotler on Marketing: How to Create, Win, and Dominate Markets. Free Press, p. 52.

Koźmiński, A.K. (2004). Management under uncertainty. Manual for advanced. PWN, Warsaw: 123-124.

Lindgardt, Z., Reeves, M., Stalk, G., Deimler, M.S. (2009). Business Model Innovation. The Boston Consulting Group, p. 2.

Maurya, A. (2012). Running Lean: Iterate from Plan A to a Plan That Works. O'Reilly, p. 18.

Maurya, A. (2014). Why Lean Canvas vs Business Model Canvas? http://practicetrumpstheory.com/why-lean-canvas/. Accessed 2014.IX.8.

Myers, S.C., Allen, F., & Brealey, R.A. (2007). Principles of corporate finance, McGraw-Hill, p. 922.

Osterwalder, A. (2004). The business model ontology. A proposition in design science approach. PhD Thesis, University de Lausanne, p. 14.

Osterwalder, A. (2014a). Value Proposition-Design. http://www.businessmodelgeneration.com. Accessed 2014.IX.01.

Osterwalder A. (2014b). https://www.youtube.com/watch?v=41q_zn8jMaE, Accessed 2014.IX.1.

Osterwalder, A., & Pigneur Y. (2010). Business Model Generation, John Wiley & Sons: 18-31.

Panfil, M., & Szablewski, A. (2009). Methods of valuation of the company. Prospective client and investor (in Polish). Poltex: 242-244.

Pucher, J. (2012). How to create a good Business Model Canvas (in Polish), http://web.gov.pl, Accessed 2012.IX.8, p. 9.

Quinion, M. (2011). Turns of Phrase: Long Tail, http://www.worldwidewords.org/turnsofphrase/tp-lon1.htm. Accessed 2014.XII.30.

Rutkowski A. (2007). Financial management (in Polish). PWE: 137-138.

Sierpińska M., & Jachna T. (2004). Rating companies using global standards (in Polish). PWN: 50, 286-287.

Teece, D. J. (2010). Business Models, Business Strategy and Innovation. *Long Range Planning*, Vol. 43: 172-194.

Welch, J., & Welch, S. (2005). Winning Hardcover. Harper Business, p. 23.

Business Models for Open Knowledge-Driven Manufacturing Execution Systems

Sari Räsänen[1] and Pavel Lederbuch[2]

[1]Tampere University of Technology, Tampere, Finland
[2]ICONICS Europe, Plzeň, Czech Republic

[1]sari.rasanen@tut.fi, [2]pavel@iconics.cz

Abstract. New ideas and technologies are commercialized and exploited through business models. It is important to search possible business models and choose the best one for the time. Unless a suitable model can be found, the technology will yield less value than otherwise it might be expected. A business model describes the rationality of how an organization creates, delivers and captures value (Osterwalder, 2010; Teece, 2010). This chapter evaluates possible business models for Open-Knowledge Driven Manufacturing Execution System (OKD-MES) platform - a next generation manufacturing execution system (MES) based on ontologies and service oriented architecture (SOA). Since the manufacturing industry is facing a transformation enabled by advancements in IT-related technologies like cloud manufacturing (Xu, 2012), the traditional business models for software business might not be suitable anymore. Business Model Canvas (Osterwalder, 2010) is used as a modelling tool for identifying and explaining the business model. This chapter provides two main contributions. Firstly, the background of business models and the main characteristics to be considered when developing a new solution to the market are introduced. Secondly, the findings are used as a basis for future research to compile business model(s) for OKD-MES.

1 Introduction

New ideas and technologies are commercialized and exploited through business models. The choice of the business model matters – the same idea or technology taken to market through two different business models will yield two different economic outcomes. Therefore it is important to search possible business models and choose the best one for the time. Unless a suitable model can be found, the technology will yield less value without utilizing the whole potential of the technology. The business model should answer for two main questions: how the company creates value for its customers (the customer value proposition) and how it captures that value (how it makes money) (Osterwalder, 2010; Teece, 2010). Business model innovation can itself offer a competitive advantage if the model is differentiated and it is difficult to replicate (Teece, 2010).

The objective of this chapter is to evaluate possible business model(s) for a next generation Open Knowledge-Driven Manufacturing Execution System (OKD-MES) - the outcome of the eScop project ("Embedded Systems for Service-Based Control of Open Manufacturing and Process Automation"), funded by the Artemis Joint Undertaking. The aim is to integrate different existing software tools for creating a solution that allows the supervisory and command capability for the manufacturing systems. The innovation of the eScop MES platform is the merging of the power of ontology-driven knowledge and service-oriented approaches (SOA) to embedded systems.

The manufacturing industry faces a transformation enabled by advancements of ICT-related technologies. The traditional business models might not be suitable anymore for manufacturing solution providers. For example, cloud computing is emerging as potentially advantageous ICT technology for the manufacturing industry. Cloud computing is seen a popular since it offers a huge space for data storage, provides flexibility and enables scalability. Therefore MES vendors have initiated developing cloud-based MES solutions to meet the demands of end-users (M2 Presswire, 2012). Xu (2012) presented two types of cloud computing in manufacturing sector, manufacturing with direct adoptions of cloud computing technologies and cloud manufacturing. The cloud manufacturing is a new way of performing manufacturing business and where everything is referred as a service – a service that is requested or a service that is provided. This Service Oriented Architecture (SOA), based on web services and web applications, creates the possibility of offering more functionality by providing end-users with much more frequent upgrades and releases without facing integration complications. This approach also raises the rationale of open-source software (OSS) based business models. It has already estimated that the availability of open-source solutions could accelerate the MES market in Europe (M2 Presswire, 2012).

Identifying and explaining the business model is often the most challenging part of communication the business ideas. The business model might be different to the different customer segments. The model needs to give clear answers what is the value for the specific customer and how the revenues are collected. Therefore different tools have been created to help visualization and modelling the business, e.g. Business Model Canvas (Osterwalder, 2010; Osterwalder and Pigneur, 2005) and Lean Canvas (Maurya, 2012).

The Business Model Canvas, developed by Osterwalder and Pigneus (Osterwalder, 2010; Osterwalder and Pigneur, 2005) (Fig. 1.) enables both new and existing businesses to focus on operational as well as strategic management and marketing plans.

The Lean canvas, proposed by Ash Maurya (2012), on the other hand, outlines a more problem focused approach and it targets entrepreneurs and startup businesses. The Lean canvas presents the current situation while the Business Model Canvas presents the end destination of the business.

The business modelling tools and the generation of a business models are introduced in more details in the two preceding chapters of this book.

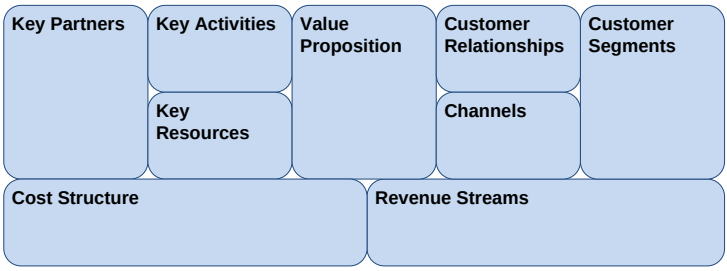


Fig. 1. The elements of Business Model Canvas (adapted from: Osterwalder, 2010)

This chapter provides two main contributions. Firstly, we introduce the background of business models and the main characteristics to be considered when developing a next generation manufacturing execution system to the market. Secondly, we use the findings from the first analysis as a basis of the final business model(s) for OKD-MES. This chapter is organized as follows: Next section introduces the background of the business models and third section presents a short analysis of first business model iteration for OKD-MES. The future steps and recommendations are discussed in section fourth. Finally, section fifth presents a brief summary and conclusions.

2 Background

The business model outlines the architecture of revenues, costs and profits. The fundamental question about a good business model asked by business strategies, is *how does one build a sustainable competitive advantage and turn a profit*? A differentiated and effective business model is more likely to produce profits (Teece, 2010). Adjusting and improving business model is a complex process and design process is likely to involve iterative steps.

A number of factors affect the selection of a business model. The competing environment, customers, financing environment, business strategy as well as the characteristics of the product and service offering sets boundaries to the models. Different business models can be viable for different situations (Rajala et al., 2003.). If one of these factors changes, the business model should be reconsidered (Teece, 2010). The major business decisions for software business includes items such as product development choices, design of product family, the level of product vs. service orientation, market development, distribution, financial, personnel arrangements and research and development procedures (Rajala et al., 2003).

The software company has several options to conduct a business. Rajala et al. (2003) identified four key characteristics of business models in software business, which enable to understand the concept of the business: product strategy, distribution strategy, revenue logic and services, and implementation. These elements help to distinguish different business models form each other and to identify the best suited model to the situation. Fig. 2. illustrates the key elements of business models, which have been combined from Osterwalder (2010), Rajala et al. (2003) and Seppänen and Helander (2014).

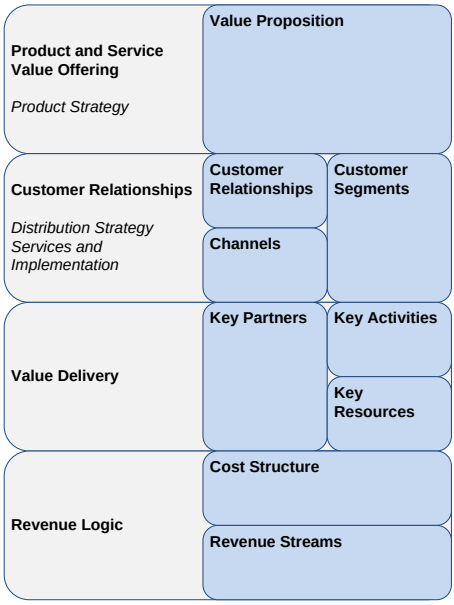


Fig. 2. The key elements of business models combined from (Osterwalder, 2010; Rajala et al. 2003; Seppänen and Helander, 2014)

Distribution strategy defines the customer relationships, the way the marketing and sales of product and service offering are organized (Osterwalder, 2010; Rajala et al., 2003; Seppänen and Helander, 2014). These are illustrated in the customer segment, distribution channel and customer relationship in Business Model Canvas building blocks (Osterwalder, 2010). Traditionally software solution providers have used their distributors or resellers to service customers with a local presence, but manufacturers today are faced with challenges. The world starts to move faster and is more connected than before. The unstable world economy, growth in consumer technology and rapidly changing consumer purchasing behaviors bring both opportunities and risks. One opportunity is the cloud that can help manufacturers reduce costs, increase agility and optimize performance. (Helo et al., 2014; Xu, 2012) Implementing cloud computing means a shift of business where everything is treated as a service, SaaS (Software as a Service), PaaS (Platform at as Service) and IaaS (Infrastructure as a Service). These three models are common types of service delivery models describing how the software is dispatched to the customer and deployed as working solution (Mell and Grance, 2010).

Value delivery describes the way the value proposition, product and service offering are delivered to the customer. These elements are described in the key activities, key resources and key partner elements in Business Model Canvas (Osterwalder, 2010). Key activities are the most important items a company must perform to make a business work. They are usually very tight to the key resources, the most important assets, and worked together. The element of key partners describes the network needed to create a value.

There are several ways of value appropriation. The revenue logic describes the sources of revenue and the way a vendor generates its profits as well as the costs incurred to operate a model (Osterwalder, 2010; Rajala et al., 2003; Seppänen and Helander, 2014). Traditionally the revenue is generated by giving customer permission to use one or more copies of the software but the ownership of the copies remains with the software vendor (so-called proprietary). With proprietary software the source code is typically commercially licensed but the open-source licensing enables anyone to further develop the original code. Open-source offers one possibility for firms to increase the amount of value. It offers opportunities for the creation of new business. Open-source software business differs from proprietary business with the type of the license, networks and their actors and the customer (Seppänen and Helander, 2014). However, the general business model usually used for proprietary software business is applicable for open-source business too. In free models, the question is how can the company offer something for free and still earn revenues? To make a profit, the company must still generate revenues somehow. One of the models is so-called freemium model that offers basic services free of charge and premium versions for a fee. This has become a familiar when the products and services are offered via Internet (Osterwalder, 2010). The companies are interested in licensing fee-free open-source business, but at the same time are concerned about the maintaining the system. This need can be fulfilled to offer stable freely available open-source software with the fee-based customer support. The customers benefit from this model since it brings the advantages of open-source software while protecting them from the uncertainties. The service provider benefits since its software is continuously improved by the open-source community, which reduces the provider´s development costs. The profit is collected as an example, as an annual fee, in which the customer has access to the latest software release with unlimited service support and the security of interacting with the legal owner of the product (Osterwalder, 2010). The companies have combined the offering of proprietary and open-source software under different licensing strategies. This is so called hybrid business model in which products, types of licenses and source of revenues are mixed. According to the empirical research of Bonaccorsi et al. (2006) hybrid models are not a transient stage but rather a permanent feature of the new industry. This allows companies to choose the best model in accordance of their customer base and product portfolios.

3 Analysis of Business Models for OKD-MES

This chapter presents the evaluation and analysis of these business models created by the eScop consortium and provides some generalization based on them. The first workshop was held in the half way of the project duration and development activities. The freedom was given to the groups to identify project outcomes and possible exploitation models. Thus the generated business models are not yet comprehensive or deployable. The main focus in this first iteration session was to brainstorm and outline the future research items.

For the design of the business model for OKD-MES the Business Model Canvas (Osterwalder, 2010; Osterwalder and Pigneur, 2005) template was used, with nine building blocks defined. The three groups containing various members of the eScop consortium created independently the business models. The outcomes of the groups are shown in the Fig. 3 and explained from the value offering, customer relationship, value delivery and revenue logic point of views. It should be highlighted that the analysis does not represent one single business model but instead several topics that should be considered. Therefore the models will be created for the different customer segments later on.

Product and Service Value Offering.

- The key values of OKD-MES solution seem to target the reduction of the system design for the automation of manufacturing systems. The most important capabilities of OKD-MES are easy and automated system configuration, agile reconfiguration of the assets when reorganizing the production plant, and reduction of the development cycle of new plant solutions.

Customer Relationships.

- OKD-MES is a solution for factory level and it should be marketed to factories, directly to factory owners or managers through system integrators that implement MES solutions. Also the identification of early adopters would be beneficial since they could bring the early feedback when using the OKD-MES.
- According to the analysis of business models for OKD-MES the direct channels like events with possible customers or customer visits should work best. Also good website presentation of OKD-MES solution, with possible online demonstrators[1], seems to be very beneficial and could help delivering the product to customers.

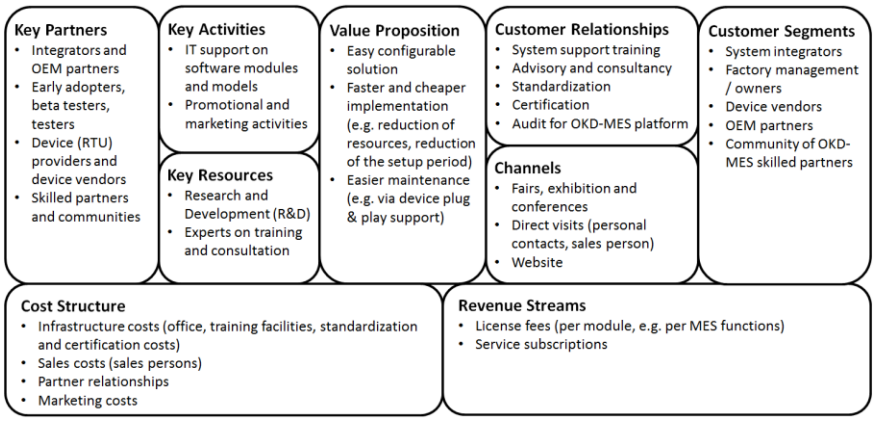


Fig. 3. The business model outcomes for OKD-MES

[1] www.escop-project.eu/teaser

- The analysis of customer relationships identified for OKD-MES shows that the close cooperation with the customers is essential. The customers should be supported by the training, advisory, and consultancy activities. Also the standardization of OKD-MES or its parts, or certification of the OKD-MES based solution would be beneficial for customers.

Value Delivery.

- R&D resources needed for modelling and configuration of the system are necessary for OKD-MES. Also subsequently customer support in the way of training and consultation experts is crucial.
- The analysis also identifies the IT support on installation, software modules set-up and customization of production models as a key activities required to be performed to make the OKD-MES business model to work. Also promoting and marketing the OKD-MES via web and application servers, various dissemination and promotional activities are necessary.
- The key partners are all the partners that could help making the OKD-MES solution working. It includes partners in all levels of the development and system integration cooperating in OKD-MES.

Revenue Logic.

- Since OKD-MES system should be based on SOA architecture the common revenue streams reflects the service oriented model of the system and fees per licensed module or per service subscription has been identified as the main revenue streams. But different revenue streams could be also considered. One of them is the model using the percentage from savings. Also revenue from providing modeling and configuration services, and extra application development could be considered. In addition, the license fee model should be clarified and open-source and proprietary model to be considered.
- The cost structure identified for OKD-MES covers the costs of the key resources and key activities required to create a value for customer. These costs include mainly the costs needed to market OKD-MES solution to customer segments.

4 Discussion

The Business Model Canvas has been a popular tool to describe the design of the value creation, delivery and revenue models. The model was used to the outline the first value chain for the OKD-MES, eScop platform. The exercise highlighted different views the OKD-MES could be exploited as well as open items to be analyzed.

The platform is directly developed to be offered as cloud based automation solution but it is still under consideration what kind of cloud deployment or service model could be implemented. These will be discussed during the second iteration.

Traditionally software solutions have been provided through distributors or resellers. Since of the market challenges, cloud based solutions can help manufacturers

tackle the economic, social or performance forces. Cloud-based solutions make possible for the companies to share common industry-specific restrictions to use a common pool of resources and best practices (M2 Presswire, 2012; Xu, 2012). It would be possible to respond to demands of multiple customers by sharing the knowledge gathered from the industry area. Thus new technological innovations should support this approach.

Data protection is the major concern of many manufacturers since it usually contains know-how of key business activities. The cloud brings new challenges in securing data and systems. The successful security policy prepares for multiple threats and perceives the different security requirements in different cloud deployment and service models (Xu, 2012). The private cloud would bring the greatest security and control, but hybrid cloud could lead the customer (manufacturer) to keep the business in the most appropriate environment. On the other hand, the cloud-based solution would let manufacturers make sure that their system is up-to-date. The deployment model should be carefully analyzed since it directly affects to chosen service model and also the customer value proposition. It is a question of trust if customers accept cloud-based solutions as public cloud (Helo et al., 2014).

The business model defines how value is delivered to the customer. At this stage the direct channels were defined as paths to customer. In addition a close collaboration was seen essential to maintain good customer relationships. These indicate that product portfolio consist not only the software product itself but also services. MES vendors have used different service models (IaaS, PaaS or SaaS) to deliver their products and services. For example Invensys, a provider of automation and information solutions to the manufacturing industry, offers some of its applications via Microsoft´s Windows Azure cloud-based platform with all delivery models (IaaS, PaaS and SaaS) (Invensys Systems, 2014). The other examples are PLEX that provides its MES functions as proprietary model with SaaS (Plex, 2014; Plex Online & Baker Tilly, 2014) and qcadoo that offers an online production management MES application for small and medium size enterprises also with SaaS. The revenue is collected from the cost of application, hosting, backup, security assurance and monitoring the servers as well as support (qcadoo, 2014b). The chosen service model will affect to the product and service offering, customer channels as well as revenue streams.

The business model answers how the value is appropriated. Is the MES software sold under proprietary license or offered as open-source solution? This is the main question since the value and thus the revenue are created in different ways via different strategies. The revenue collected from proprietary licenses has been the traditional strategy. Since it has been estimated that the availability of open-source solutions could accelerate the MES market in Europe (M2 Presswire, 2012), and since open-source could offer new opportunities for the creation of new business, the format of license model must be evaluated during the next iteration of eScop business models. The main reason for open-source model would be to get the product widely adopted. A wide adoption increases the probability of attracting professional developers and achieving a higher pace of technology development (West, 2003). A widely diffused product can also get first-mover advantages – such as setting technological standards or attaining a substantial market share (Dahlander and Magnusson, 2005). Since many

research innovations are poorly protected, it requires novelty from business models to bring positive spin offs (Teece, 2010).

Seppänen and Helander (2014) introduced in their research how the use of a business model enables value creation in an Open-Source Software (OSS) environment. They provided a list of questions from every Business Model Canvas (Osterwalder, 2010) elements that can help managers to decide the implementation of OSS strategy. The main question concerns the architecture of software: *Does the software architecture allow for pieces, or even the full package, of software to be open-sourced?*

Kilamo et al. (2010) introduced the framework to determine how complete a software is for releasing in open-source and what its potential to attract external developers is. In addition, freemium business model is widely deployed by software companies who also operate in the open-source markets. The basic form of software is licensed under open-source and the premium version with additional features and services is available under commercial license (Teece, 2010).

Next step is to evaluate the proprietary and open-source strategies for the OKD-MES software. The open-source strategy for MES solutions is in initial stage and at the moment there are not many free solutions available. For example, qcadoo has a community edition application available in their webpage with a minimal set of features to solve a single problem in the production line (qcadoo, 2014a). On the contrary, there are many open-source Enterprise Resource Planning (ERP) software available in the market, e.g.: (Compiere, 2013) and (ADempiere, 2014).

5 Conclusions and Future Work

The developments in global economy require businesses to re-evaluate the value proposition they offer to customer. Technological innovation does not assure business or economic success (Teece, 2010) - the choice of the business models matters. It is affected by the constraints set by the competing environment, customers, resource environment, business strategies as well as the characteristics of the product and service offering. The core of a business model is to define the way the company delivers value to customer and converts the customer payments to profit (Osterwalder, 2010; Teece, 2010). The analysis of the business model canvas building block provided in the previous section shows that the key factor when marketing and selling OKD-MES is the reduction of the resources (costs, time, personnel, etc.) for the implementation and maintenance of the MES solution. This leads to the conclusion that the OKD-MES solution should be offered to factory management as a solution that should save the money and time in the phase of factory setup and its maintenance.

The findings from this analysis are used as a basis of the final business model(s) for OKD-MES. The product and service offering needs to be defined and value proposition to be clarified independently to the different customer segments. This also requires more detailed analysis of customer relationships, distribution strategy and service delivery implementation. Since the revenue strategy might be different to the different customers, the licensing strategy needs to be considered – how the value and thus the revenues would be created.

References

ADempiere (2014). ADempiere EPR Homepage. http://www.adempiere.com/ADempiere_ERP. Accessed 2014.XI.7.

Bonaccorsi, A., Giannangeli, S., & Rossi, C. (2006). Entry Strategies Under Competing Standards: Hybrid Business Models in the Open Source Software Industry. *Management Science*, 52(7): 1085–1098.

Compiere (2013). Compiere Homepage. http://www.compiere.com/products/open-source/index.php. Accessed 2014.XI.7.

Dahlander, L., & Magnusson, M. G. (2005). Relationships between open source software companies and communities: Observations from Nordic firms. *Research Policy*, 34/4: 481–493.

Helo, P., Suorsa, M., Hao, Y., Anussornnitisarn, P. (2014). Towards a cloud-based manufacturing execution system for distributed manufacturing. *Computers in Industry*, 65/4: 646–656.

Invensys Systems (2014). Invensys Homepage. http://iom.invensys.com/EN/pages/home.aspx. Accessed 2014.XI.7.

Keahey, K., Tsugawa, M., Matsunaga, A., & Fortes, J. A. B. (2009). Sky Computing. *IEEE Internet Computing*, 13(5): 43–51.

Kilamo, T., Aaltonen, T., Hammouda, I., Heinimäki, T. J., & Mikkonen, T. (2010). Evaluating the Readiness of Proprietary Software for Open Source Development. *IFIP Advances in Information and Communication Technology*, Vol. 319: 143–155.

Lawton, G. (2008). Developing Software Online with Platform-as-a-Service Technology. *Computer*, 41(6): 13–15.

Maurya, A. (2012). Running lean: iterate from Plan A to a Plan That Works. O'Reilly.

M2 Presswire (2012). Research and Markets: Manufacturing Execution System Market in Europe 2011-2015. *M2 Presswire*, Coventry, 2012.XI.6.

Mell, P., & Grance, T. (2010). The NIST Definition of Cloud Computing. *Communications of the ACM*, 53/6: 50–50.

Osterwalder, A. (2010). Business model generation: a handbook for visionaries, game changers, and challengers. J. Wiley & Sons.

Osterwalder, A., & Pigneur, Y. (2005). Clarifying business models: origins, present, and future of the concept. *Communications of the Association for Information Systems*, Vol. 16: 1–25.

Plex. (2014). Plex Homepage. http://www.plex.com/industries-and-solutions/manufacturing-execution-system.html. Accessed 2014.XI.7.

Plex Online & Baker Tilly (2014). Six Game-Changer about SaaS ERP: Plex Online Brings a New Approach to Manufacturing Software. http://www.bakertilly.com/uploads/Six-game-changers-about-SaaS-ERP.pdf. Accessed 2014.XI.7.

qcadoo (2014a). Qcadoo Community Edition Homepage. http://qcadoo.com/community-edition-download.html. Accessed 2014.XI.7.

qcadoo (2014b). Qcadoo Homepage. http://qcadoo.com/. Accessed 2014.XI.7.

Rajala, R., Rossi, M., & Tuunainen, V. K. (2003). A Framework for Analyzing Software Business Models. *ECIS Conference Paper*.

Seppänen, M., & Helander, N. (2014). Creating Value through Business Models in Open Source Software. *Int'l Journal of Open Source Software and Processes*. Vol. 5/2: 40-54.

Teece, D. J. (2010). Business Models, Business Strategy and Innovation. *Business Models*, 43(2–3): 172–194.

West, J. (2003). How open is open enough?: Melding proprietary and open source platform strategies. *Open Source Software Development*, 32(7): 1259–1285.

Xu, X. (2012). From cloud computing to cloud manufacturing. *Robotics and Computer-Integrated Manufacturing*, 28(1): 75–86.

Holistic System of Evaluation of Production Systems Efficiency

Stanisław Marciniak

Warsaw University of Technology, Warsaw, Poland

iosp@wip.pw.edu.pl

1 Introduction

This chapter presents a proposal of efficiency evaluation methodology for integrated production systems and equipment, assuming that the production environment includes advanced manufacturing systems and advanced solutions for production planning and control. Irrespective of its comprehensiveness and objectivity, such an evaluation should aim not only at controlling or performance management purposes. It should also stimulate relevant reconfiguration of production system architecture, according to changing needs and requirements, e.g. through improvements that exploit the ICT or automation innovations.

The contemporary approaches to integration of production systems are typically based on intensive utilization of novel ICT and automation technologies (Lobov et al. 2009). For instance, the eScop approach is based on the combination of:

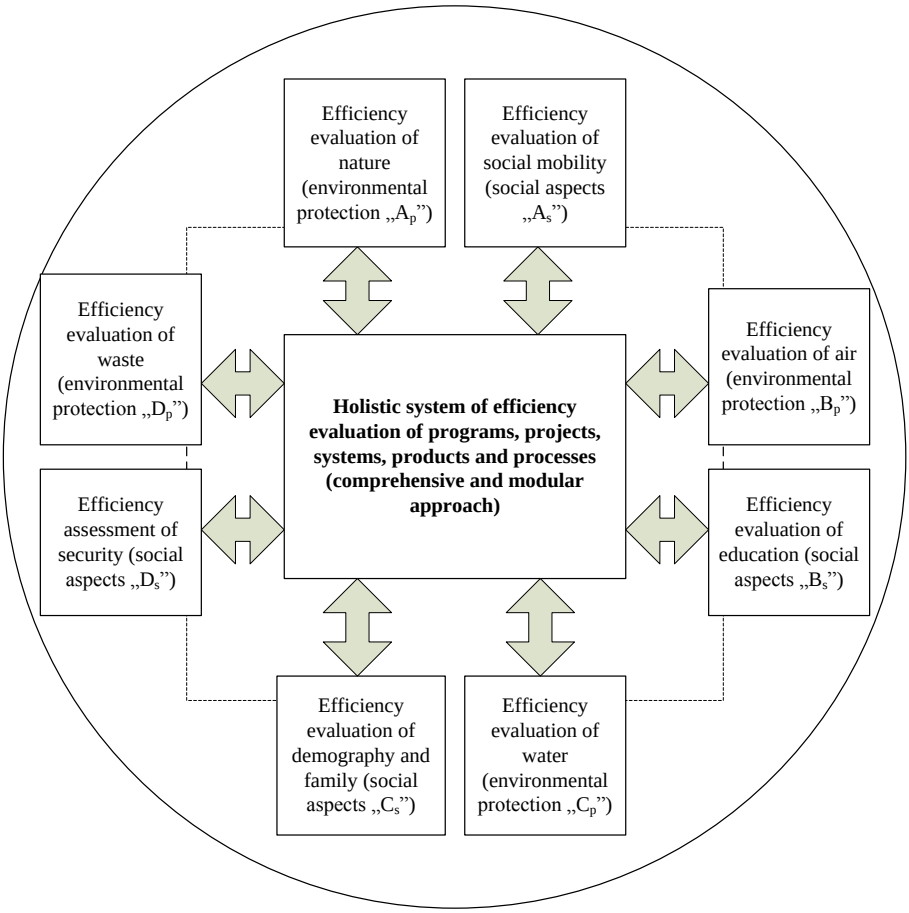
1. application of the embedded devices and appliances,
2. knowledge-based management supported by the means of ontology engineering ,
3. application of the SOA (Service Oriented Architecture) and Web-services.

The principle of open knowledge-based manufacturing, based on the means of ontology engineering, is particularly adequate when preparing the evaluation of efficiency of integrated production systems and equipment (Garetti and Fumagalli, 2012). It is due to its universality and holism, which push towards the preference for holistic efficiency evaluation, i.e. with respect to contemporary paradigms and rules of designing and implementing evaluation systems.

The holistic evaluation system should be both, open and comprehensive. The comprehensiveness is reflected by the feature that evaluation process directly encompasses the whole of different aspects (e.g. economic, technological, environmental), and it is done not just by assessing the impact of a given item only on strictly understood economic evaluation. The complexity of production system is reflected not only in its integration, but also in the distinctive hierarchy of structural elements, which should respect the criterion of their influence on the holistic evaluation. Structural ties in modern production systems may be equally technical (physical), like also organizational, environmental and social.

2 The idea of holistic system of efficiency evaluation

The holistic system of efficiency evaluation (Fig. 1) is mostly based on the paradigm of new economy (Marciniak and Głodziński, 2010; Godłów-Legiedź, 2010), which advocates that the efficiency evaluation process should encompass strictly economic, as well as the social and environmental phenomena.



The aggregated holistic evaluation of projects

where:

environmental protection	social aspect
A_p - nature	A_s - social mobility
B_p - air (atmosphere)	B_s - education
C_p - water	C_s - demography and family
D_p - waste	D_s - security (health service, pensions, public security).

Fig. 1. The Integrated Holistic System of Efficiency Evaluation (IHSEE)

The area of applications for such a system encompasses: programs, projects, systems, products and processes, of different level of technical advancement. All these items require wide perspective of analysis concerning their development, implementation and exploitation (Sanjay, 2008). Therefore, the system can and should be also applied to evaluate programs, projects, systems, products and processes in all areas of technically intensive services, like the ICT services, energetics, agro-food processing and medical services (Grasso and Martinelli, 2010).

The structure of the system includes a formal description of the modular approach to the evaluation in the three mentioned areas. The dynamics of such a system is supported through control loops and feedbacks, which take place along functioning of the modular mechanism for identification of dependencies (MID), including these existing between the whole modules of measures (along the first stage of the evaluation), and also those between specific measures representing particular modules (i.e. along the second stage of evaluation).

The idea of holistic system for efficiency evaluation of projects is based on relevant design, development, implementation, and exploitation of four structural elements, which are later on presented one by one in the following figures. They include:

1. the integrated holistic system of the project efficiency evaluation (IHSEE),
2. the methods of economic efficiency evaluation (EEE),
3. the methods of social efficiency evaluation (SEE),
4. the methods of environmental efficiency evaluation (EcoEE).

which constitute a whole, and create a comprehensive and integrated system.

The proposed holistic system of programs, projects, systems, products and processes efficiency evaluation, should follow the laws and premises of sustainable development which are currently promoted within the economies of most developed countries, and particularly by the EU. The requirements resulting from the holistic system of efficiency evaluation are also being imposed on countries in the process of development.

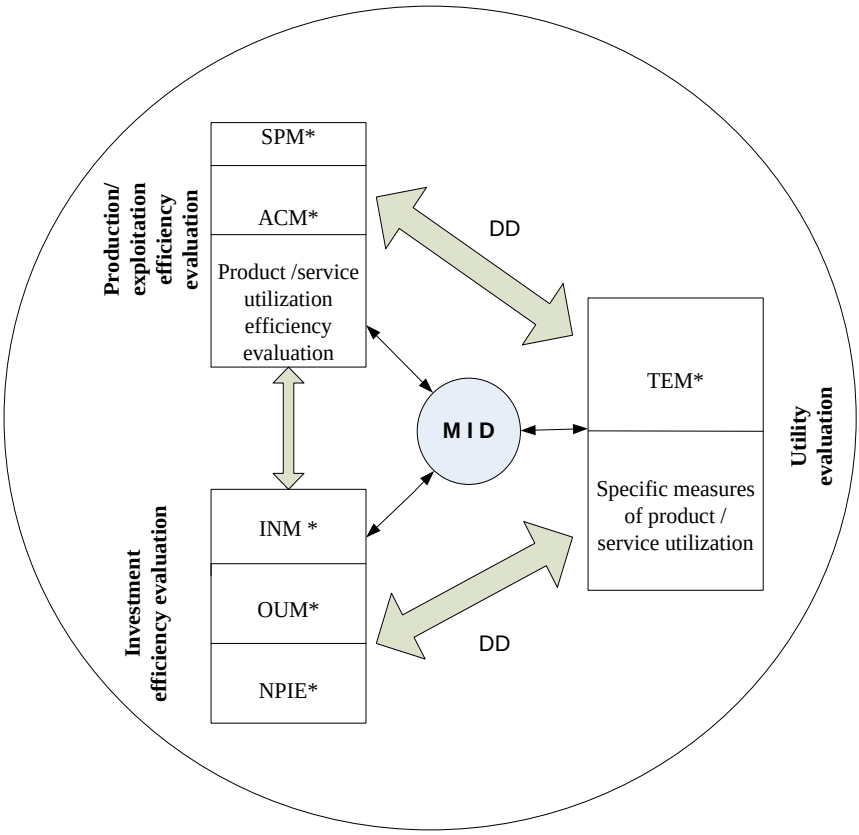
It should be underlined, that according to the proposed principles, the holistic efficiency evaluation encompasses all aspects which affect the living conditions in a given country, according to its specific level of development.

3 The holistic structure of the efficiency evaluation system

The structure of holistic efficiency evaluation system encompasses three elements:

- the evaluation of economic efficiency (EEE) (Fig. 2),
- the evaluation of social efficiency (SEE) (Fig. 3),
- the evaluation of environmental efficiency (EcoEE) (Fig. 4).

The integrated character of evaluation means that evaluation process in the same time targets some clearly defined domain (e.g. company activities, production process, etc.), while conducting the evaluation by a set of ordered and coherent (i.e. not inconsistent and not correlated) measures.

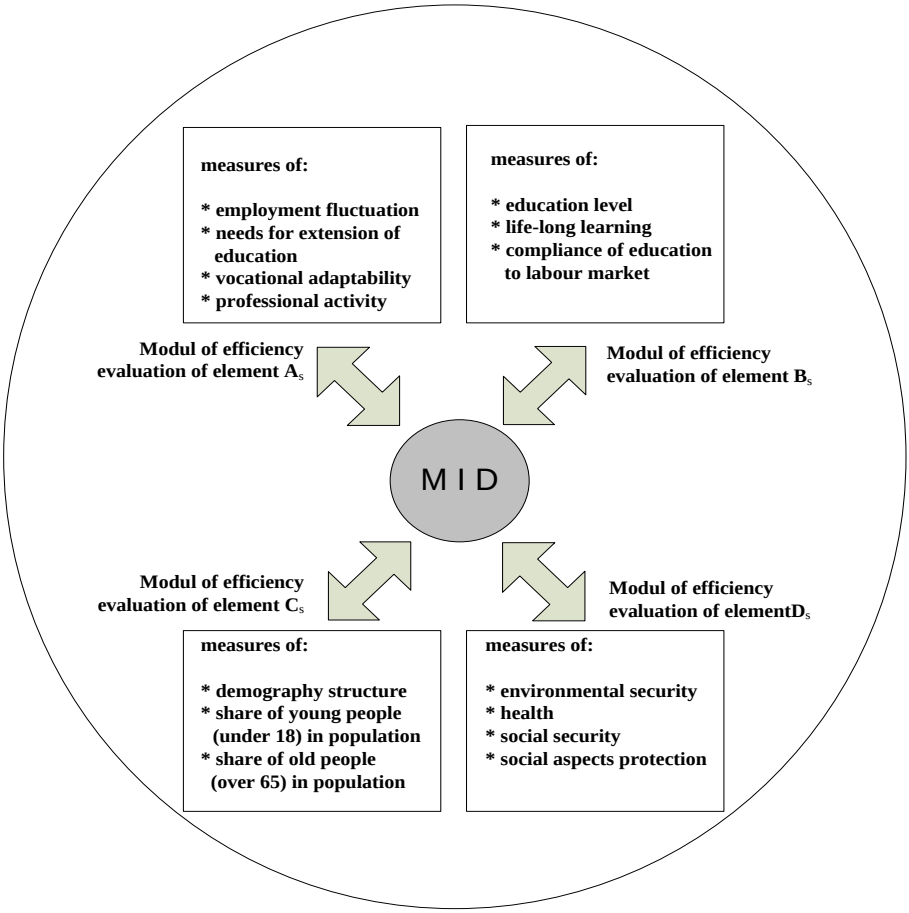


Comprehensive and integrated evaluation

where:
SPM – simple partial measures, ACM – aggregate/complex measures, INM – input measures, OUM – measures of output (effects), NPIE – non-preferable evaluation of investment, TEM – technical/exploitation measures (parameters); DD - direct dependencies; MID - mechanism for identification (or definition) of dependencies between measures representing particular modules (mostly regarding indirect dependencies)

Fig. 2. The structure of economic efficiency evaluation as a component of the holistic evaluation system

The widely understood economic efficiency evaluation of an economic system addresses those economic activities that are realized in a specific area (a country or a region, etc.), and along of a particular period of time.

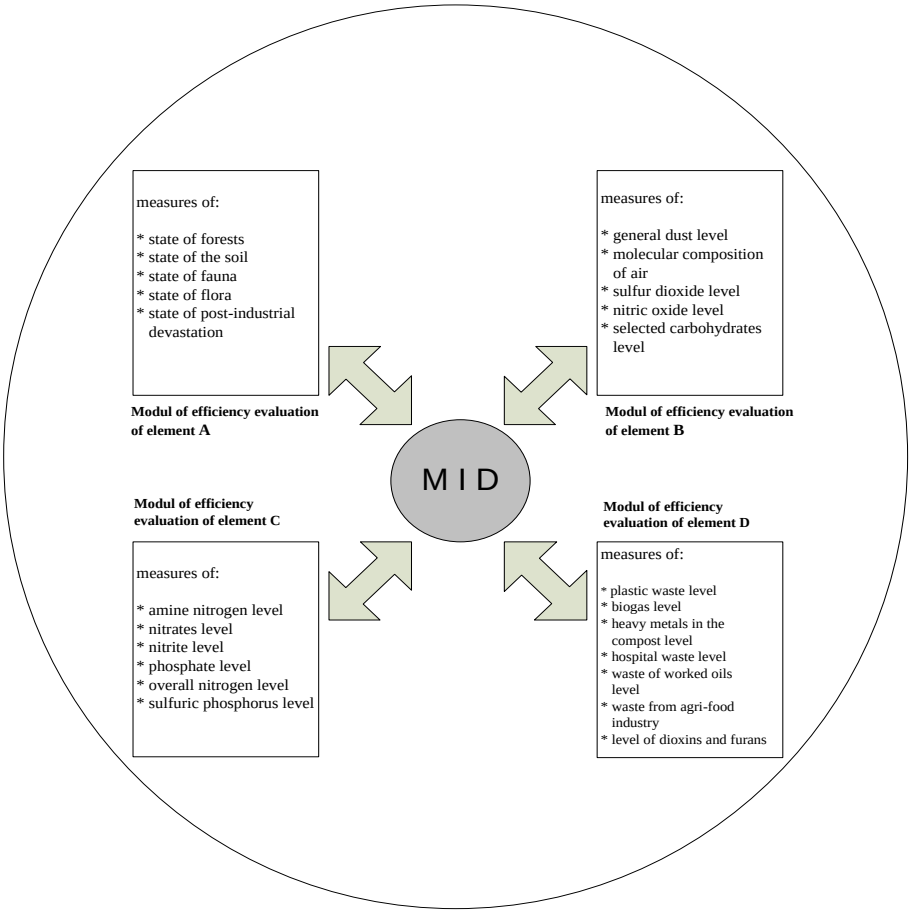


Comprehensive and integrated evaluation

where:
MID - the mechanism for identification (or definition) of dependencies between measures representing particular modules (mostly regarding indirect dependencies)

Fig. 3. The structure of social efficiency evaluation as a component of the holistic efficiency evaluation system

The evaluation of social efficiency of programs, projects, systems, products and processes encompasses different types of social description concerning the human factor. It is used to represent those changes of social characteristics, which result from implementation of modern technological and organizational solutions.



Comprehensive and integrated evaluation

where :
MIZ - the mechanism of dependency identification (definition) between metrics representing particular modules (regarding mostly indirect dependency)

Fig. 4. The structure of environmental efficiency evaluation as a component of the holistic evaluation system

The evaluation of the environmental efficiency of the programs, projects, systems, products and processes encompasses different types of environment description items. It is used to reflect changes in the environment characteristics which result from implementation of modern technical and economic solutions.

4 The holistic system of efficiency evaluation vs. time

The time aspect of holistic efficiency evaluation is normally considered when taking two different perspectives:

- time as the moment of (phenomenon) evaluation,
- time as the factor (or time interval) for characterizing the changes of phenomenon.

According to the principles of systems theory, the accepted and adapted efficiency evaluation system, e.g. for a given production system, should be compatible with the management system. Time should be one of the dimensions which ensure compatibility and consistency of evaluation. Management systems, regardless of our preference for centralized and hierarchical or distributed and flat, should be evaluated in a relevant way for a given phenomenon. It should be done in both possible cases, i.e. in a snapshot way (the moment of evaluation), as well as by evaluation of the trend characterizing given phenomenon for a defined and specific period (timeframe).

The choice of the snapshot evaluation point depends on:

1. legal and financial requirements, mostly regarding reporting standards,
2. organizational and legal changes of the business entity (e.g. regarding property),
3. dramatic (groundbreaking) technical and technological changes in the functioning of the production process.

The trend evaluation results from:

1. market changes, sequential and turbulent in time,
2. the implementation of innovative solutions, regarding both products and processes,
3. dramatic financial and legal changes.

The trend evaluation should also consider the changes of goals of given business entity, which are defined in the company strategy.

Within the evaluation of the advanced production systems efficiency both approaches towards time-based evaluation should include:

1. the suitability evaluation for realization of tasks concerning external communication of information,
2. efficiency of internal communication with regard to the production system,
3. suitability of tools (techniques) applied in production management.

Risk level is the parameter, that is mostly related to the time dimension of any economic phenomenon. The risk determines (or limits) efficiency of any given activity. This topic is explored to more details in the section 6.

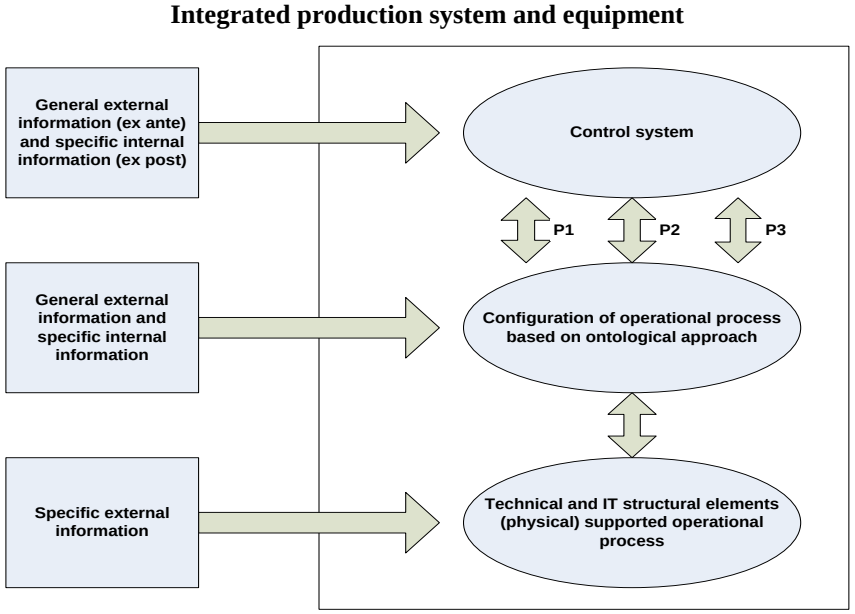
5 The holistic system of efficiency evaluation vs. information

To be properly implemented, the holistic system of efficiency evaluation should meet a number of requirements concerning the information that is needed along its opera-

tion. From the perspective of decision-making, the input information should be comprehensive and differentiated according to the specific requirements concerning evaluated programs, projects, systems, products and processes. Within the frames of contemporary complex production systems, the support information should encompass three perspectives:

1. Controlling,
2. Configuration of operational processes (e.g. production),
3. Structural (physical) elements enabling appropriate operation of the system.

The range and type of necessary information at the level of these three segments is presented in Fig. 5.



p_1, p_2, p_3 – feedbacks in various phases of process configuration of the system (process)

Fig. 5. The information characteristics of the efficiency evaluation system

The presented range of information is aligned with the principles of eScop approach. However, some conditions have to be respected to enable meeting the goals that are set for information support (Marciniak, 1989). This problem was examined more deeply by Warnecke and Braun (1999), who identified the features of the information systems, which are expected to support management and exploitation of open and complex production processes that intensively use advanced manufacturing technologies. The cited authors claim that such systems should support:

- diagnosis of the activities in terms of requested needs,
- the selection, screening and acquiring information and knowledge from different sources,
- processing the information into task-oriented forms, which may be defined depending on their users,
- generation of solutions,
- making decisions according to presumed priorities,
- further processing of the selected options of solutions,
- dispatching information to other organizational units.

The fulfillment of above mentioned requirements by an information system is particularly necessary for efficient operation of open complex productions systems, i.e. those that are expected to be supported by the eScop approach. On the other side the paradigm of open knowledge driven manufacturing that is grounding the eScop approach is expected to meet the listed requirements.

6 The principles of designing and operation of the holistic system of efficiency evaluation

The efficacy, and consequently, practical suitability of the efficiency evaluation system, often depend on proper definition of design principles and operation of the system. Following the principles seems to be more important, when the consequences of a mistake at this stage, are difficult and costly to amend in the operation phase. The principles may be divided into universal and specific.

The universal design principles include the protection of:

1. comprehensiveness,
2. sequencing of activities,
3. relevant information,
4. adequate time and risk parameters.

The specific design principles include the protection of:

1. hereditary characteristics,
2. comprehensiveness,
3. being up to date,
4. internal and external communication.

The universal principles of operation include:

1. continuous monitoring of the evaluation system operation,
2. possibility to quick react to the changes,
3. flexibility with regard to internal and external change,
4. compatibility with the direct and indirect environment.

The specific principles of operation include the possibilities of:

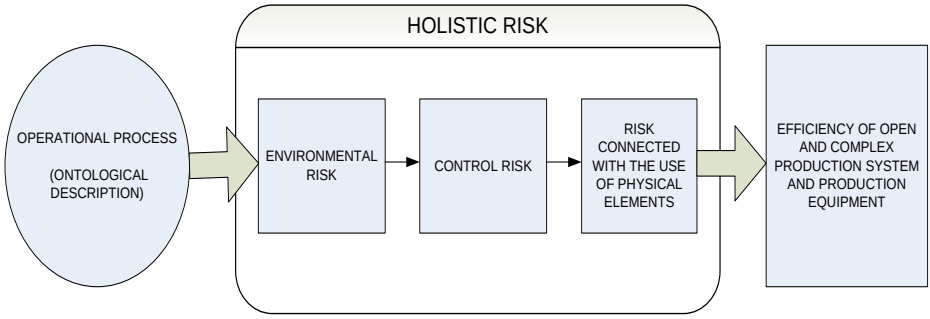


Fig. 6. Risk vs. system efficiency

1. system modularity regarding the characteristics of evaluated item,
2. limitations of structural complexity,
3. communication improvements,
4. definition of evaluation hierarchy and values of partial assessments.

The principles listed above are coherent. It is necessary to follow all of them in order to protect objectivity and efficacy of the whole set.

When presenting the principles of design and operations of the holistic system of efficiency evaluation it is vital to define the influence of holistic risk on the system efficiency. Fig. 6 illustrates this dependency.

The risk level of any kind included in the holistic risk frames depends on the time period of the system efficiency evaluation. The longer the time horizon of analysis of the solution, the more important is the external risk. The shorter the time, the more important is the risk of functioning of physical elements of the process (facilities and equipment).

7 Case study - Holistic efficiency evaluation of integrated production systems in reference to the eScop approach

The presented framework for holistic efficiency evaluation of integrated production systems can be adopted to the particular case of those production systems, that follow and implement the eScop approach. This subchapter introduces in a brief way a proposal of such an system of evaluation (Table 1). It directly follows the structure of evaluation, as introduced within the preceding subchapters. This means that it encompasses the three perspectives of evaluation, as well as the modular compositions of measures of evaluation for all of them.

The proposal presumes that evaluation of a given production system may be implemented in three different contexts, all of them in relation to the exploitation, extension or modification of a given production system. One of them is an ongoing, repetitive process of business planning. Herein the proposal is to be exploited while using a

particular relevant management instrument, e.g. the controlling, therefore it is primarily used for performance management in reference to some manufacturing system.

Table 1. Holistic efficiency evaluation of production systems using the eScop approach

	Module / submodule	Measure
Economic persepctive	Production system reconfigurability	Lead time of redesign Lead time of reconfiguration
		Cost of redesign Cost of re-certifications Cost of revalidation Cost of reconfiguration
	Short-term demand flexibility	Lead times
		Unit cost
	Production system performance	Lead times Work-in-progress Utilization Outages
		Unit cost
	Investment efficiency evaluation	Standard measures
Social per-spective	Unemployment	Job sustainability ratio
	Job/post safety	
	Employment safety	Job retention ratio
	Work safety	Reduction of accidents Reduction of losses due to accidents
Sustainability perspective	Environment safety	Waste / emissions Reduction of waste Reduction of emissions Critical/catastrophic events Effects of critical/catastrophic events Losses due to critical/catastrophic events
	Energy consumption	Consumption of energy Reduction of energy consumption
	Natural resources consumption	Consumption of renewable resources Reduction of renewable resources consumption Consumption of non-renewable resources Reduction of non- renewable resources consumption

The second possible implementation of the proposed evaluation scheme is along assessment of a particular reconfiguration of some production system. In such case it is expected to support the decision process in reference to reconfiguration of production systems understood as an investment in its facilities. Finally, the proposal can be used within the scope of feasibility and acceptability assessment of introducing eScop

based solutions to an existing production system. This means that particular effects of performance of the production system are focused on, as the subject of its evaluation.

The production systems, that follow and implement the eScop approach are expected to provide particular performances. Among them the reconfigurability is given a particular attention. The eScop approach was primarily developed with respect to the needs and requirements that have to be met by the European industries in reference to the changing and diversifying market expectations. Apart of that, short term flexibility of production systems is of particular importance. Hence those performance characteristics, that are correlated to the short term efficiency and effectiveness of acceptance of the variable demand, are taken into account, namely: lead times, work-in-progress, utilization. The improvements of them are expected to be achieved by advantages of the knowledge driven shop floor control (or OKD MES), e.g. by a better coordination of operations, by early warning of possible outages and so on (Strzelczak, 2014). Due to the flexibility trade-off the unit costs are also considered as aggregate measures of different aspects of productions system economic efficiency.

The social impacts by production systems, that follow and implement the eScop approach are expected in a threefold way. The improved competitiveness and performance of production system should result in a better job safety of its employees. Apart of that the employment in such a system requires and supports better skills. Hence the employment safety of any employee should be also improved. Finally, better abilities to monitor production processes and the system should result in a better work safety. With regard to this effect a similar but secondary effect can be named. As suggested later on in this subchapter, the production systems which follow and implement the eScop approach are featured with a better ability to protect against critical or catastrophic events. These may eventually have negative health impacts, or can even bring and exposure to the peoples' life. Hence, the eScop based solutions may eventually help to reduce such negative effects and the losses due to them, like penalties and compensations.

The sustainability impacts by production systems, that follow and implement the eScop approach are exhibited in a threefold way. The eScop type solutions provide a better ability to monitor production processes with regard to waste emissions, energy consumption as well as consumption of renewable and non-renewable natural resources. Secondly, due to exploitation of the knowledge based approach, i.e. of some kind of limited intelligence, process optimizations and optimized process controls are eventually possible, that may result in reduced waste emissions, energy consumption as well as reduced consumption of renewable and non-renewable natural resources. Finally, the improved abilities of monitoring production processes may help to avoid critical or even catastrophic events, which possibly outcome in some kind of environmental pollution.

The presented proposal is rather limited, although fully valid. It was intended as an illustration of the discussed approach in reference to the case, which is of special attention in this chapter. An example of holism of this approach is visible in the mentioned interdependencies between improved process monitoring and environmental and social impacts. The framework can be easily extended or further adopted to some exact needs and requirements.

8 Advantages and disadvantages of the holistic system of efficiency evaluation

The assessment of advantages and disadvantages of the method resumes the presented theoretical framework, which should be further verified in the economic practice.

The advantages result mainly from the character of the method and they include:

a) Comprehensiveness, which directly supports its holistic character,
b) Easy way for measuring complex processes,
c) Sequencing of the evaluation process (three levels, i.e. application of the principle: holistic-modular-measurement-based approach),
d) Openness, which allows direct consideration of the social and environmental factors,
e) high flexibility and adaptability, which allows to preserve compatibility with the evaluation systems functioning in the real conditions.

Disadvantages, just like advantages, are connected with the paradigm of adopted method. They include the following difficulties:

a) in the evaluation of specificity of the solution,
b) in obtaining proper details of the evaluation,
c) assessment of structural elements of the evaluation, from the perspective of possibilities to implement the method.

The above mentioned characteristics have been only partially discusses in the literature (Pham et al., 2008). However, they form a kind of whole, which supports the following thesis: in the current conditions of developed economies, which accept paradigms of the new economy, it is necessary to apply holistic evaluation of efficiency and efficacy. It particularly regards novel solutions, which will be applicable in the future, e.g. in the area of automation, ICT, bioengineering, i.e. in line with the presumptions of eScop project. These kinds of solutions should not be evaluated only from the economic perspective according to the rules and principles that were valid in the 20th century. It should be changed according to the paradigm regarding the scope of assessment, considering the external impacts. Furthermore, the way of thinking about technical and organizational solution, while decision taking concerning a particular option, should be also changed.

9 Conclusions

The holistic system of assessment of production system was presented in this chapter, which is applicable to those systems that are automation and ICT intensive. It is actually an updated and modified version of the integrated method of efficiency assessment of the technical and organizational projects, which was originally developed for the flexible manufacturing systems domain. It was redesigned by taking into account

contemporary management knowledge, especially in reference to the concept of sustainable economy.

The idea of presented holistic system for efficiency evaluation was presented herein, followed by structural approaches to the modules of assessment, and finally by ideation of comprehensive integrated assessment. The relations between holistic system and the time and information related aspects were also described. Finally, principles of design and operation of the holistic efficiency assessment system, like its strengths and weaknesses, were also discussed.

The application of the presented system follows contemporary management concepts and enables managers taking better decisions regarding efficiency and effectiveness of manufacturing systems and processes.

Acknowledgement: This work has been co-funded by the Grant from Warsaw University of Technology (PSP 504/01518/1103/40.000105).

References

Garetti, M., & Fumagalli, L. (2012). P-PSO ontology for manufacturing systems, *Proceedings of the 14th IFAC Symposium on Information Control Problems in Manufacturing*: 247-254.

Grasso, D., & Martinelli, D. (2010). Holistic Engineering, In D. Grasso, & M. Brown Burkins (Eds), *Holistic Engineering Education. Beyond Technology*, Springer: 11-16.

Godłów-Legiedź, J. (2010). Contemporary economics. Coming to a new paradigm? (in Polish). C.H. Beck.

Lobov, A., Ubis Lopez, F., Villasenor Herrera, V., Puttonen, J., & Martinez Lastra, J.L. (2009). Semantic Web Services Framework for Manufacturing Industries, *Proceedings of the IEEE International Conference on Robotics and Biomimetics* (2009): 2104-2108.

Marciniak, S. (1989). The integrated method of efficiency assessment of technical and organizational projects (in Polish). Warsaw University of Technology Publishing House.

Marciniak, S., & Głodziński, E. (2010). Influence of new economy on the development of company management control tools (in Polish). In J. Pyka (Ed.), *Novelty of industry and services - models, methods and tools for managing organizations*, TNOiK: 229-243.

Pham, D.T, Pham, P.T.N., & Thomas, A. (2008). Integrated production machines and systems - beyond lean manufacturing, *Journal of Manufacturing Technology Management*, Vol. 19/6: 695-711.

Sanjay, B. (2008). Lean and performance measurement. *Journal of Manufacturing Technology Management*, Vol. 19/5: 670-684.

Strzelczak, S. (2014). Ontology-Aided Management, *Silesian University of Technology Series in Management*, 73(2014): 619-630.

Warnecke, H.-J., & Braun, J. (Eds) (1999). From Fractal to Manufacturing Network: the creation of efficient company cooperation. Springer.

PART TWO

IMPLEMENTING ONTOLOGIES

Implementing Ontologies in Manufacturing and Logistics

From Theoretical Fundamentals to Prospects

Stanisław Strzelczak

Warsaw University of Technology, Warsaw, Poland

s.strzelczak@wip.pw.edu.pl

Abstract. Ontologies enjoy growing interest as the means for semantic integration of: information, models, applications, appliances, systems and ecosystems. The focus of this chapter is planning and controlling operations. A special attention is given to those abstractions and formalizations that ground ontologies for open knowledge-driven manufacturing and logistics. Considering such specific qualities of the future operational environments, like complexity, variability and dynamics, an expressive and efficient meta-ontology may decide about both, the efficacy of planning and control solutions, and the non-turbulent flow of operations. With this regard theoretical enhancements and extensions to the existing approaches are presented herein. They are based on the transformational paradigm, which was developed at Warsaw University of Technology. It applies the architectural construct of 'Tasks-tRANSFORMations-rEsouRceS' to abstract about such systems and ecosystems, which encompass the interacting and coupled: demands, operations and capacities. Following that, the TRANSFORMERS ontological framework is introduced. Although it was primarily targeted to support planning and control of operations, it also enables other important functionalities, like the qualitative simulation and gamification of systems and ecosystems. This way anticipation and assessment of behavioral qualities, including the emergent ones, is made possible. Important examples could be the turbulent behavior or the alienated self-sufficient distributed intelligence.

1 Introduction

Manufacturing and logistics operations are recently exposed to the rapid growth of openness and networking. The second momentum effects from that are increased complexity and variability. Simultaneously intensive innovations in the areas of ICT and automation technologies (AT) strongly affect the progress in the domain. Among the key novel technologies there are: Web services, semantic technologies, smart appliances, Big Data technology, Cloud computing, tracking technologies et al.

The most distinctive areas of development can be referred as seven Internets (Fig.1). The existing Internet of data, which is expected to evolve towards the Semantic Web (Berners-Lee et al., 2001), provides a public infrastructure for information exchange to the other Internets. Some of them are pretty advanced and undergo intensive development, like especially the Social Internet and the Commercial Internet.

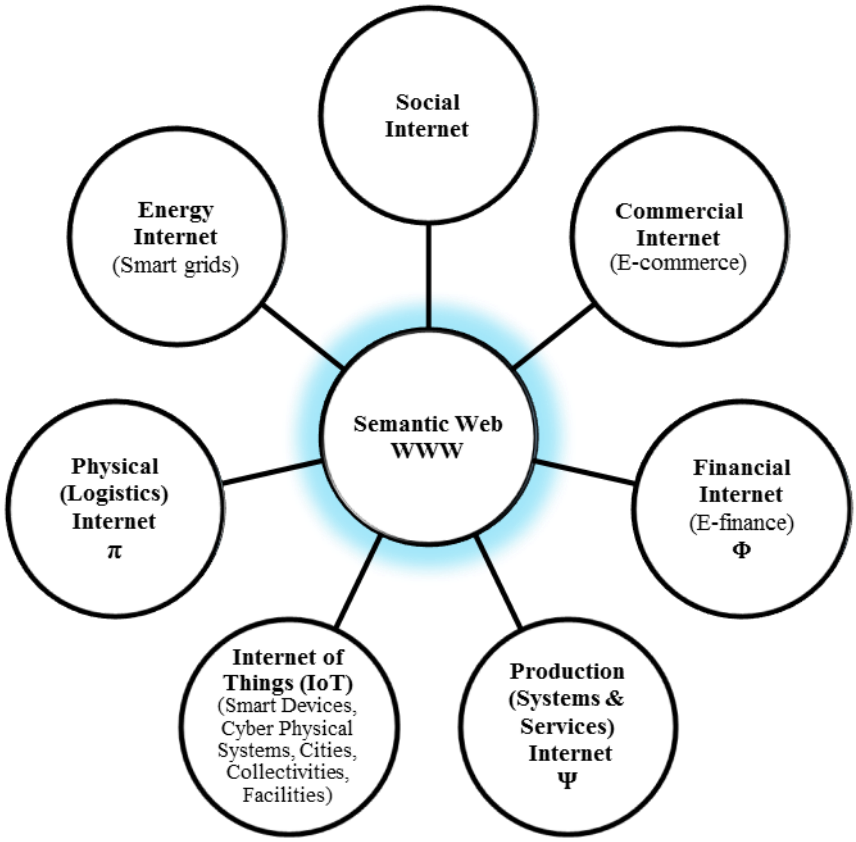


Fig. 1. Semantic Web driven Seven Internets

Other Internets are still more ideations than reality. Nevertheless, significant efforts towards their development are undertaken and considerable support is offered by the funding bodies, as all of them were put to the top of research agendas.

The concept of **π**, i.e. Physical (Logistics) Internet, extends the idea of encapsulation of data from the Internet to logistical flows (Montreuil, 2011). The moving entities would be physically encapsulated in the modularized **π**-containers (Fig.2).

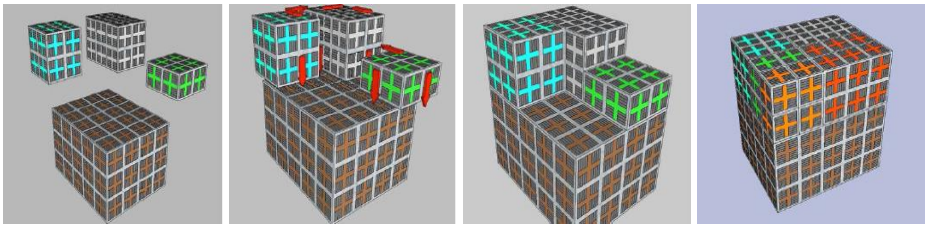


Fig. 2. Encapsulation of goods in **π**-containers (based on (Montreuil, 2012))

The transport of goods would be operated by the worldwide standardized smart means, including **π**-containers, those stored in **π**-stores, handled by **π**-movers or **π**-conveyors (Fig.3), transported by multi-modal network of networks composed of **π**-routes (Fig.4) and **π**-nodes (Fig.5).

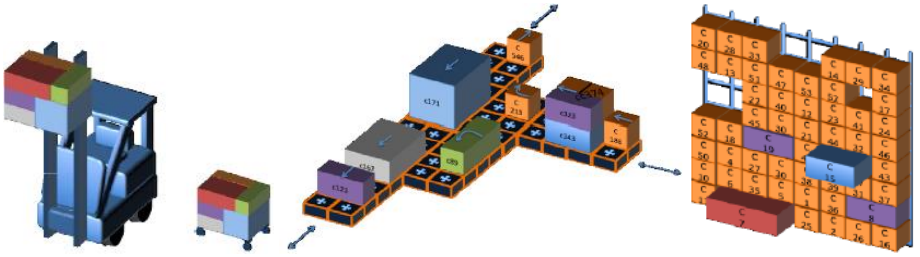


Fig. 3. π-movers, **π**-conveyors and **π**-stores
(based on (Montreuil, 2012))

Fig. 4. FoodTubes and CargoCap infrastructural projects of encapsulated demand http://www.ilookforwardto.com/2010/12/foodtubes-really-fast-food-delivered-in-a-physical-Internet-of-underground-pipes.html http://www.cargocap.com/content/what-is-cargocap

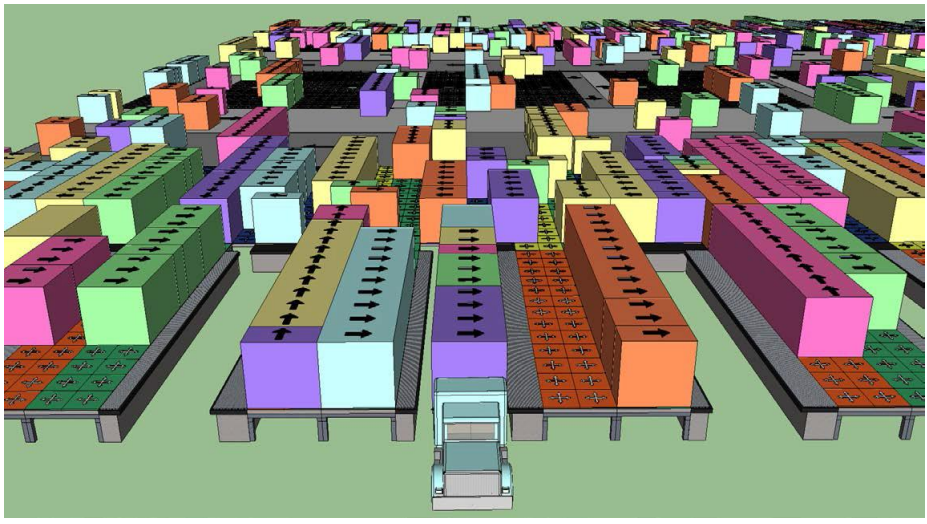


Fig. 5. π-node - Road Hub of π-containers truck-to-truck cross-docking
(from (Montreuil, 2012))

The idea of **π** extends to human mobility. An encapsulated pneumatic supersonic transportation of humans is the subject of development of the Hyperloop project, which is driven by Elon Musk (Fig.6). The **π**-based human mobility would be supported by the means of open infrastructure, which would cover both, the local commuting and distant mobility of people. As another example development towards that direction the solution of Uber Technologies can be recalled, which presumes a new way of transportation in agglomerations. It is called 'urban logistics fabric' and assumes occasional involvement of drivers, who controlled by smartphones can shuttle passengers together with picked up items along their way.

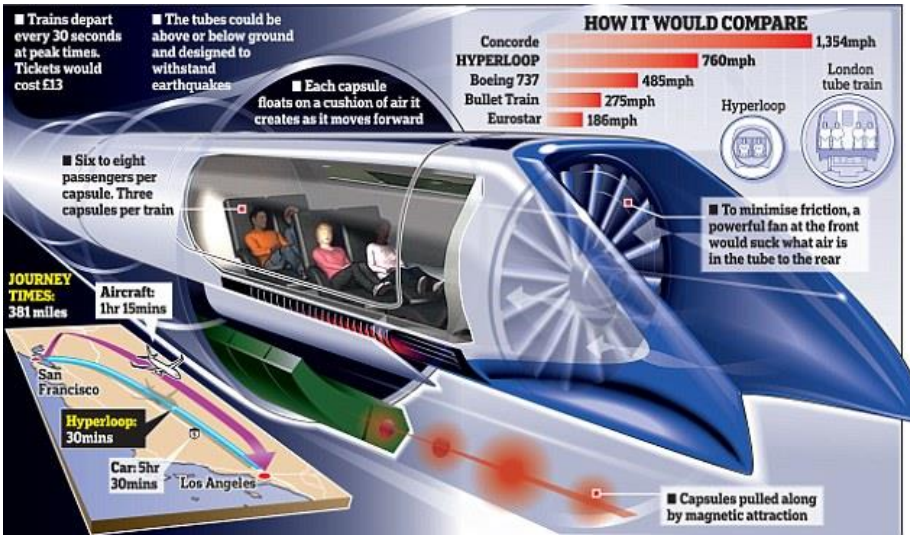


Fig. 6. The Hyperloop project of encapsulated transportation of humans; by Elon Musk (http://www.dailymail.co.uk/sciencetech/article-2912170/LA-San-Francisco-half-hour-moves-closer-Elon-Musk-reveals-plans-Hyperloop-test-track-Texas.html)

The intention of **π** is to improve the way, how physical objects are moved, stored, realized, supplied and used throughout the world. Nowadays it is economically, environmentally and socially inefficient and unsustainable (Montreuil, 2011). Enabling seamless, fast, cheap, safe, reliable, as well as distributed, multimodal transport and deployment of π-containers across the Physical Internet, it is expected to reduce or eliminate many weaknesses of the existing solution, namely:

1. *Improve the efficacy of logistics*: The global transport efficac is below 10% (Ballot and Fontane, 2008). In UK the proportion of truck-kilometers travelled empty is 27% (McKinnon, 2007). In USA 20% of all miles driven are with a completely empty trailer (Ballot and Fontane, 2008), with many more nearly empty. Carriers are often half-empty at departure, while a large part of the non-emptiness is filled by packaging (Levinson, 2006). Vehicles and containers often return empty, or incur extra travel routes to find return shipments. Furthermore, vehicles leaving loaded get emptier and emptier as their route unfolds from the point of origin to the destination. Similarly, storage facilities are poorly utilized.

2. *Improve the life of truckers - the modern cowboys* (Montreuil, 2011): Continental transportation is road-based. This causes a high demand for drivers. Truck drivers are the people who are nearly always on road. They have no chance for normal personal, family and social life. Being overloaded, they cause most of the on road accidents (Johnston III, 2015). The **π** is expected to bring the life of drivers back to normality (Fig.7).

Fig. 7. Effects of **π**: improved economic performance, work conditions, human & family life (from (Montreuil, 2012))

3. *Improve the rotation and availability of materials and goods*: Materials and goods stay mostly in store, being stored where unneeded and often unavailable fast where needed. A significant portion of consumer products e never reach customers on time, ending up unsold at some location, while being required elsewhere. Although statistics rarely report this issue, it is well known in the food and clothing industries, and also happens with high-price products such as cars, anecdotally evidenced by rusting new cars in a disused airfield (Montreuil, 2011).
4. *Avoid unnecessary transportation and crisscrossing the world* (Montreuil, 2011): Materials and goods travel around the globe. This could be avoided by smart routing or processing them close to their point of use. The offshoring has significantly enhanced this phenomenon. Even without it, the hub-and-spoke networks and centralized distribution systems, with centers covering large areas, lead to the excessive transportation.
5. *Overload and vulnerability of transportation networks*: An extreme concentration of operations in a limited number of agglomerations and industrial & commercial zones, with goods travelling along a small number of overloaded roads, make the transportation networks both, inefficient and vulnerable (Peck, 2006). In case of extreme events the negative effects might be enormous and long lasting (Tainton and Nakano, 2014). This issue is especially sensitive in big agglomerations, as getting products in, through and out of big cities became a nightmare in recent decades (Crainic, 2008). Additionally, noise and pollution affect the citizens.

Interestingly, the idea of **π** follows the architecture of public logistical infrastructure developed by the state of Incas. Although the carrier resources were only human and animals, the efficacy of transportation was impressive for today standards, including the speed of forwarding. The system of Incas was employing the concept of hubs and localized use of resources, similarly to the idea presented in Fig.7. A widespread access and availability of the infrastructure to everyone provided particular advantages. The goods produced locally could be widely distributed without any own effort. Therefore the whole economy could be spatially distributed, being highly integrated. The economic forces did not drive towards huge agglomerations, as its spatial spread did not act against overall efficiency. Some of the logistical concepts developed by the Incas, which helped to make their economy flourishing, could be possibly mimicked by the **π**. With this regard good examples are the transportation by herd carriers, which recalls the Internet of data, or the system of vertical archipelago for redistribution of goods (Rostworowski de Diez Canseco, 1999). The logistics infrastructure of the Incas state was the key factor of welfare and comparative advantage of the whole economy, as well as the key pillar of the military power of the state.

The **Ψ** (psi, i.e. the Production Systems & Services Internet) has been originally described in (Strzelczak, 2014c). **Ψ** is expected to match and align the demand coming out of the Commercial Internet, with the available provisions offered by the networked production systems services (Fig.8). The bulking and encapsulation of the demand, and of the resource provisions, would be integrated by the means built upon Semantic Web, while the resources of **π** and **Φ** would support in an integrated manner the operations of **Ψ** towards the customers.

The development of Production Internet has to face a particular difficulty. While the other Internets operate mostly on a peer-to-peer basis, herein the links between demand and supply flows are subjected to various couplings (Fig.9), which normally exhibit the nature of spatiotemporal mereotopological relationships. Therefore harmonizing the two streams requires an additional coordination support, which could be some kind of Web service.

Enabling smooth, fast, cheap, reliable alignment of distributed demand and manufacturing resources, by merging its capacities with **π**, the Production Internet is expected to reduce or eliminate many shortcomings of the existing economies, namely:

1. The economic order quantities of logistics flows and manufacturing batches can be exploited in parallel.
2. The matching of demand and manufacturing provisions can be operated in a more flexible way. Instead of time-stress based planning and control, as it takes place nowadays, the matching of demand and operations can be done within the applicable time-limits, e.g. as exhibited by the subscribed demand.
3. Additionally, spatial matching of demands and provisions can be exploited by the means of **π**, building the synergy of both Internets this way.
4. Following above the overall utilization of manufacturing and logistics resources can be much improved, for the health of economy, society and environment.
5. Operations can be streamed and streamlined, while manufacturing activities can be servitized.

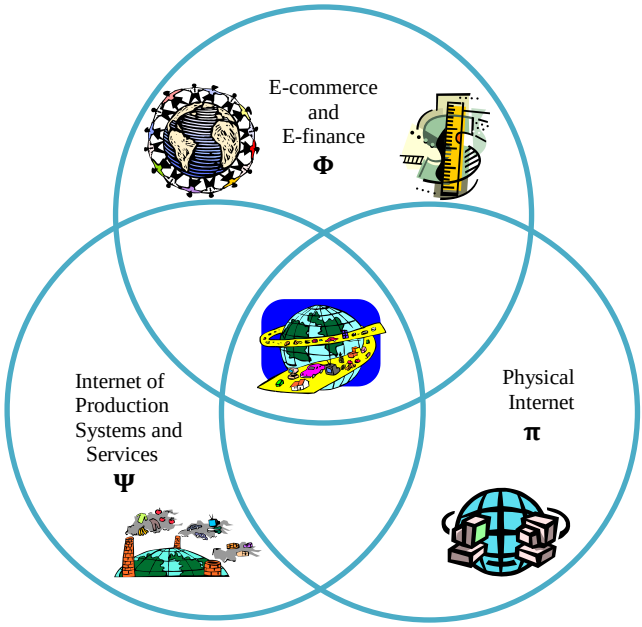


Fig. 8. Positioning of Production Internet

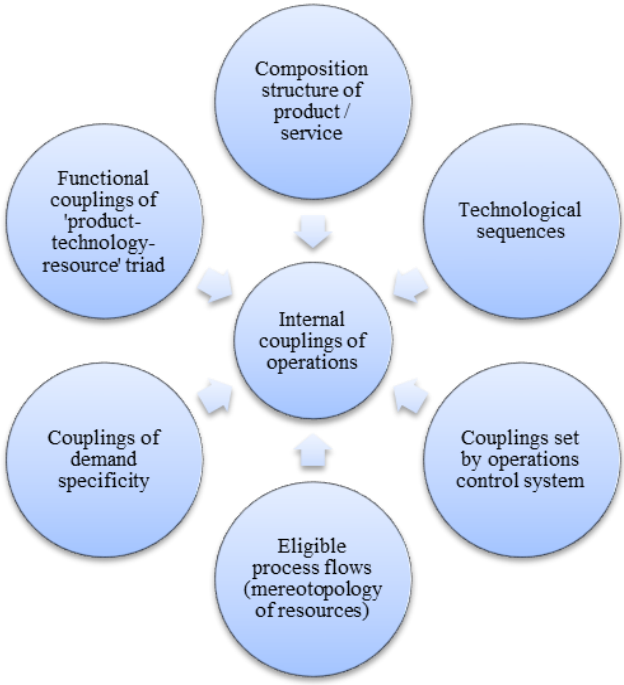


Fig. 9. Internal couplings of operations

6. The distortion of demand and turbulent behavior (e.g. with regard to the bullwhip effect), will be significantly reduced.
7. The transaction costs economy can be tamed, giving more space to the open market-like economic institutions.
8. Therefore, the damaging impacts of monopolies and hierarchies (i.e. in terms of institutional economics) can be much reduced.
9. The economic forces that push towards geographically concentrated economy, can be much reduced. Hence, the local communities can be hopefully revitalized.
10. The economies will be less turbulent and less vulnerable, particularly with regard to the possibility of extreme events.

Notably **Ψ** exhibits a novel form of economic institutions, offering advantageous alternative to supply chains, industrial networks, sector clusters and monopolies.

Some existing Web services can be considered as the heralds of **Ψ**:

- Groupon as an example of a demand bulking service;
- Zuora as a demand subscription service;
- NetSuite OneWorld as an example of a Cloud service for multi-company management of resources;
- Kickstarter crowdfunding service as an example of demand, supplies and funds amalgamation service.

The Internet of Things (IoT) aims at enabling ubiquitous connection of physical devices equipped with the smart connective technology (RFID, sensors, cameras, GPS, Internet, etc.), capable of generating and collecting data by themselves, capacitated with the distributed self-control through the networked networks of IoT (Höller et al., 2014).

Accompanying concepts are also discussed in the literature, like the Industrie 4.0 (Sendler, 2013), which assumes future industrial systems to be built upon so called CPS (Cyber-Physical Systems) (Lee, 2011). The Cyber-Physical Systems are commonly understood as complexities of embedded systems integrated into various devices and facilities that are eventually endowed with the Human-Machine Interfaces (HMI).

The social and technical potential of semantic integration has been deeply explored by some domains, e.g. by the healthcare sector, or recently in the area of social networking (Feigenbaum et al., 2007). Ontologies are recently recognized as the most appropriate mean for knowledge representation, if used to support the semantic processing and various knowledge-driven functionalities, especially in such fields, like the medical informatics (Rosse and Mejino Jr., 2003), geographic information processing (Fonseca et al. 2003), and software engineering (Aβmann et al., 2006).

It is often argued in the literature that major challenges and issues arising along the development of seven Internets can be resolved by implementation of ontologies (Chandrasekaran et al., 1999; Lobov et al., 2007; Strzelczak, 2014b). In accordance with this argumentation the use of ontologies in manufacturing and logistics attracted attention of many scholars (Garetti and Macchi, 2003). Various functionalities and aspects are addressed by authors, like: manufacturing and supply chain planning and

control (Strzelczak, 2014a), shop floor level control (Fumagalli et al,. 2014), reconfiguration of manufacturing systems (Garetti and Fumagalli, 2012), product life-cycle management (Matsokis and Kiritsis, 2010), ontology-based product modelling and design (Sun et al. 2010), et al. A lot of attention is also given to the knowledge-based integration of industrial automation systems by the means of Web services (Ramis et al., 2014), and agent-based systems (Vrba et al., 2011).

Ontology engineering was intensively developed in recent years, mostly due to the efforts of Semantic Web initiative driven by the World Wide Web Consortium (W3C). The Semantic Web is expected to provide an open infrastructure for Web services and intelligent agents. It presumes ontologies as knowledge representations. The Ontology Web Language (OWL) together with other standards was defined with this regard (Hitzler and Janowicz, 2011). Ontology engineering provide principles, methods and tools to support creation and exploitation of ontologies.

It was exposed in chapter 2 that the published ontologies for the manufacturing and logistics domain merely provide high level taxonomies. The aid to manage and control activities, like other ontology-aided functionalities, are addressed to a very limited extent, if at all. The same applies to dynamical aspects and complexities. Therefore research and practice gaps exists.

Ontologies can work as common platforms for interacting applications. Hence, they can be used for various purposes, i.e. going beyond the scope of planning and control, by aiding different management and control activities. E.g. conceptualizations used for the latter functionalities, are also useful for the operations strategy process. When assessing acceptability of any operational process, a holistic estimation of its performance characteristics is normally required (Slack and Lewis, 2011). Similarly, diversified estimations are typically required within the feasibility evaluation. In both cases the ontological representation of a given system, i.e. as developed for the planning and control purposes, can support the requested functionalities.

Ontologies can merge qualitative and quantitative representations of knowledge and data. Hence, they enable qualitative simulations or gamifications of operations, which can facilitate assessment of qualitative characteristics, like the behavioral ones. E.g., it is possible to assess the impact of rules and behaviors of agents on the turbulent or idiosyncratic behavior of the operational ecosystems (Strzelczak, 2012).

It was argued in chapter 2 that an extended perspective on ontology-aided in manufacturing and logistics, can help to ideate, conceptualize and facilitate novel functionalities and advantageous systemic solutions. However, some formal extensions and modifications of the grounding abstractions are likely to be required with this regard.

The above questions will be explored more or less directly, and in-depth, within the consecutive sections.

This chapter continues considerations of chapter 2 about ontology-aided manufacturing and logistics. Novel systemic solutions and functionalities presented therein are confronted with the existing capacities of ontology engineering. The key question discussed herein is about the relevance of theoretical foundations, and particularly of those ontological abstractions that ground existing formalizations of ontologies, like the ontology languages. A special attention is given to those aspects of operations management, that relate to complexity, dynamics and variability. They are identified,

reviewed and systemized. With this regard spatiotemporal, mereological and mereotopological couplings are focused on, which arise along the planning and control. It is argued that some theory-based enhancements and extensions to the existing ontological approaches are indispensable, otherwise a limited efficacy and effectiveness may limit both, the advantages of new solutions in terms of novel capacities and improved performance.

Section 2 reflects on theoretical foundations of ontologies, which are primarily supported by the meta-mathematical and logical theories: mereology, ontology and protothetic. Then section 3 reflects on the existing means of ontology engineering and discusses some particular advantages, i.e. with regard to those needs and requirements which are crucial for the manufacturing and logistics domain. Section 4 investigates dynamic complexities that normally arise along operations management. They are reflected as layered and more or less granulated interdependencies, which can relate the entities subjected to planning and control. Then section 4 reflects on the Service Oriented Architectures (SOA), Web services, and multi-agent systems. It is analyzed if the founding concepts of these paradigms and frameworks fit well enough to the domain interest. The conclusions are subsequently used in section 6 to argue that a particular threefold construct should ground the abstractions of ontology used to aid planning and control of manufacturing and logistics operations, i.e. the 'Tasks-tRANSFORMations-rEsouRceS'. Consequently the TRANSFORMERS ontological framework is introduced in section 7, which is built upon of the transformational triad. It is consistent with the SOA paradigm and can support the multi-agency, while being more generic and expressive. Later on an ontology-based framework for qualitative simulations and gamifications is presented in section 8. It enables to discover and assess behavioral characteristics of operational systems and ecosystems, e.g. the turbulent behavior. The chapter is summarized and concluded in section 9.

2 Theoretical foundations of ontologies

This section reflects on theoretical fundamentals of ontologies. A special attention is given to complexities which arise during planning and control activities of manufacturing and logistics operations. With this regard a review of relevant meta-mathematical theories is presented, which is supported with some insights from the theory of categories (Adamék et al., 2004).

The questions concerning meta-mathematical and logical foundations of ontologies were attracting the interest of researchers long ago, and it was long before the ontologies were adopted by the areas of ICT and Artificial Intelligence (AI). The theory that directly founds contemporary ontologies was developed by Stanisław Leśniewski – the Polish logician and philosopher, who along the 1910/1920s tackled in a comprehensive way problems around the Russel's paradox, and then he proposed a complete and holistic system composed of three meta-mathematical theories, which are:

- Protothetic: calculus of phrases (propositional calculus);
- Ontology: calculus of names (calculus of classes);

- Mereology: theory of part-whole (calculus of parthoods).

The whole system is based on two fundamental categories, i.e.: propositions and names. This is Leśniewski who actually coined the term ontology in the contemporary sense, i.e. as it is used and understood by the ontology engineering. The composition of Leśniewski's system is depicted in Fig.10.

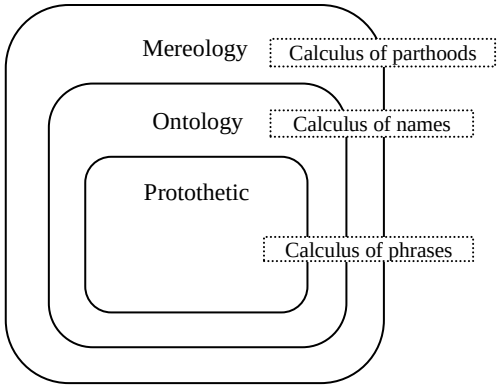


Fig. 10. Leśniewski's system of meta-mathematical foundations for formal ontologies

Later on the works of Leśniewski have been continued by his followers and among them, by his only PhD student Alfred Tarski, who is ranked as top-four logician in the history of humankind (together with Aristotle, Frege, and Gödel), in early 1930s defined the concept of semantic truth and formalized principles of logical semantics (i.e. theory of axiomatic models) and its relations with syntactics (Tarski, 1944).

Prototheticc uses equivalence as the sole primitive term and propositions that can be combined by (1 ÷ n)-ary functors, to form new propositions. Expressions can be built from propositional variables, sentence forming functors, a universal quantifier, any constant of any properly defined semantic category (using the formerly introduced symbols), and variables belonging to a previously introduced category.

Actually prototheticc is a generalized sentential calculus (Srzednicki and Stachniak, 1998). Because the principles of bivalence and extensionality are among its theorems, it also establishes in some sense an absolute sentential logic, with quantifiers binding:

- (i) sentential variables;
- (ii) functorial variables ranging over functors forming sentences of sentential arguments;
- (iii) functorial variables ranging over functors of functors; etc.

Quantifiers in the theorems of prototheticc bind variables of any semantic category which are definable when starting with a category of sentences. Prototheticc is consistent, but the problem of its completeness is not yet fully solved. However, it was at least proven that the elementary prototheticc, i.e. with quantifiers restricted to sentential variables is complete.

Interestingly, by using the theory of protothetic it can be proved that **axioms of ontology do not determine the meaning of primitive constants**, which is the commonly accepted claim (Urbaniak, 2013). The last comment exposes a significant limitation to the reasoning capacities of contemporary ontologies, like those expressed in OWL. This amplifies a question about the full relevance of major promises given by the Semantic Web initiative (Marshall and Shipman, 2003), and possibly reveals a significant factor of its still limited dissemination[1]. As Tim Barners-Lee stated together with his colleagues referring to the Semantic Web "… This simple idea … remains largely unrealized. … " (Shadbold, 2006).

Surprisingly, although the formal power of protothetic is significant, it has not been widely adopted, and it is still rarely known to the researchers, except some universities that lead in the area (e.g. Stanford University, Columbia University). Actually the founding ideas behind prototethic are not easy to understand, which applies even – as it is visible in some publications - to logicians.

Ontology was aimed as an axiomatized theory of common names and functors of an arbitrary order. Using the functor 'is' (denoted as **ε**) Leśniewski proposed as a foundation to the ontology a single axiom, to enable forming sentences from names and defining subsumption hierarchies. The axiom was originally expressed as follows:

{A **ε** a} if and only if ((for some B, {B **ε** A}) and (for any B and C, if {B **ε** A} and {C **ε** A}, then {B **ε** C}), and (for any B, if {B **ε** A} then {B **ε** a}))

On the basis of this axiom Leśniewski developed a powerful system of a general logic comparable in strength to the simple type theory. He named this system 'ontology' because he regarded it as a logical theory which offers a general theory of objects in the Aristotle sense (Srzednicki et al., 1984). Ontology allows definition of different sentence-forming functors operating on names. It performs the role of predicate logic, but important differences exist between them, e.g.: identity is definable in a first-order ontology, whereas it is not definable in the standard first-order logic, and the theorems of ontology are true in any domain.

The first entire mereology as the theory of parthood was completed by Leśniewski in 1916 (Srzednicki et al., 1984). It was using as the basic functor 'part'. The parthood relation is understood as a relation of part to whole or a relation of part to part within a whole. The parthood relation is reflexive, transitive and antisymmetric, i.e.:

1. Everything is part of itself;
2. Any part of any part of a thing is itself part of that thing;
3. Two distinct things cannot be part of each other.

[1] A similar case can be recalled from the story of hydrogen bomb development. The original solution, rooted in physical intuitions was proposed by Edward Teller. In 1951 a young Polish mathematician Stanisław Ulam approached him and suggested that his idea is not valid and will not work. Ulam referred to his own mathematical model. Teller rejected Ulam's comments, as he actually did not understand them. However after few months of fruitless efforts he changed his mind and got back to Ulam. The merged ideas of two men, based on the Ulam's model, resulted in a final successful concept known as the Ulam-Teller design.

The above can be regarded as axioms for forming a first-order theory, assuming the use of a standard first-order language with identity, supplied with a binary predicate constant ‘part’, and taking as the underlying logic the predicate calculus with identity. Such theory is often referred as the ‘Core Mereology’.

The crucial idea of the mereology is hidden in the definition of mereological sum (sometimes called fusion or aggregate), which respects the following equivalence:

an object S is a mereological sum of the group of s-es if and only if every s is part of S and every distinguishable part of S is compatible with some s

The above opposes mereology as a theory of collective classes to the theories of classes built upon the standard set theory (e.g. subsumption hierarchies are such distributive classes), which considers classes as collections of elements satisfying a given property. The key difference between these conceptions is that elementhood in mereology is transitive, while membership in set-theory is not. Actually Leśniewski himself has shown the departure of parthood relation from the core set-theoretical assumptions, which was principally his intention. This may explain that the common theories of knowledge representation operate basing on the subsumption relation, while the parthood relation is clearly not known or not recognized (Mormann, 2012). In many sciences, like biology, ecology and the environmental sciences, the subjects of considerations exhibit the mereological morphology, but not the set-theoretic one. The distributive structures are common and welcome by the ICT community, which enjoys the simplicity of object-oriented and similar approaches, while in the area of manufacturing and logistics even the static view of reality does not hold such presumptions. Therefore the theoretical contribution by Leśniewski is fundamental.

Various concepts of portioning can be used to interpret, or alternatively to supplement or found different mereologies, like: overlap, underlap, ingredient, mixture, composition, containment, extension. In principle relata of ‘part’ can be as different as material objects and substances, events, states, geometric entities or spatiotemporal regions, as well as abstract entities such as properties, propositions, types, kinds etc.

An important extension to the mereology was proposed by Clarke (1981), who suggested to add the primitive ‘connection’. Therefore the mereological concept of parthood was integrated with a quasi-topological component, concerned with connection This way a founding theory was established, coined later as the mereotopology.

The basic concepts of mereotopology refer to relations between wholes and parts, parts, parts of parts and borders between parts (Smith, 1996; Casati and Varzi, 1999). The three axioms of core mereology are supplemented by the three additional mereotopological axioms. Assuming ‘connection’ as a second founding primitive, i.e. apart from the ‘part’ one, these additional axioms can be formulated informally as follows:

4. Everything is connected to itself (reflexivity);
5. If a thing is connected to another thing, the second one is connected to the first thing (symmetry);
6. Everything is connected to anything, to which its part is connected (monotonicity).

Using the various concepts of the connection relation different mereotopologies can be designed to facilitate qualitative spatial reasoning in a requested way (Cohn and Varzi, 1998). Examples of such topological concepts are: boundary, contact, separation, interior and exterior. All of them typically refer to the concept of closure and its standard axiomatization used in topology, i.e. the one defined by Polish mathematician Kazimierz Kuratowski (1922). Consequently, whenever the spatial aspects of phenomena exhibit some features of topological structures, mereotopology can be considered as an advantageous alternative to those theories of classes, which are founded upon the standard set theory.

The topological extension of mereology added a significant capacity to the original theory. The basic conceptualizations in many sciences clearly refer to meretopological abstractions. This is also the case of manufacturing and logistics. The expressions of demand that are commonly used along demand processing, can be given as an example. The conceptualization of paths and modules in an automated manufacturing systems is another one example. Herein the importance of connection primitives is manifested, as it is not only important to express how the resources are structured, i.e. in the sense of the theory of sets or systems, but also how they are connected by the paths of material flows or control flows. The two examples explain the significance of theoretical contribution given by Leśniewski, and then extended by Clarke.

Mereotopologies focus on parthood and connection relations in spatial regions. In many domains, like manufacturing and logistics, spatial reasoning involves reasoning about entities that may coincide without overlapping. Mereotopologies typically do not offer capacities to support distinction between objects or processes and regions in which they are located, which can be at best described by appealing to distributions of attributes across the occupied regions. The distinct relation of spatial coincidence, which focuses on sharing of location, is normally not introduced.

With this regard the theory for layered mereotopology was developed (Donelly, 2003). It adds to the core mereotopology relations of relative location, such as the coincidence which solely depends on the locations of objects. Also the relation between a spatial entity, and the region at which it is located, is made explicit.

All existing definitions of coincidence in the layered meretopology refer to the ‘overlap’ relation. Two entities overlap when they share a common part. By one of the common definitions two entities coincide when they occupy overlapping regions of space. In such case unary function is added which assigns each object to its region, i.e. at which the given object is exactly located. Using such function a one-place predicate, which distinguishes the sub-domain of regions can be defined. Alternative definition of coincidence avoids the concept of region, hence in most cases it fits better to the requirements of manufacturing and logistics: two objects coincide if they are both contained by another object. Consequently it is possible for two objects, to exist simultaneously in the same location. In other words it is acceptable that coincident objects do not share parts, not like in the core mereotopology. In region-free mereotopologies the function assigning objects to regions is not needed. Instead the ‘contain’ or similar relations are used (Donelly, 2004). Layers are non-overlapping partitions. Layer is a sum of all objects that it underlaps. Members of the same layer coincide only when they overlap. A simple illustration is presented below in Fig.11.

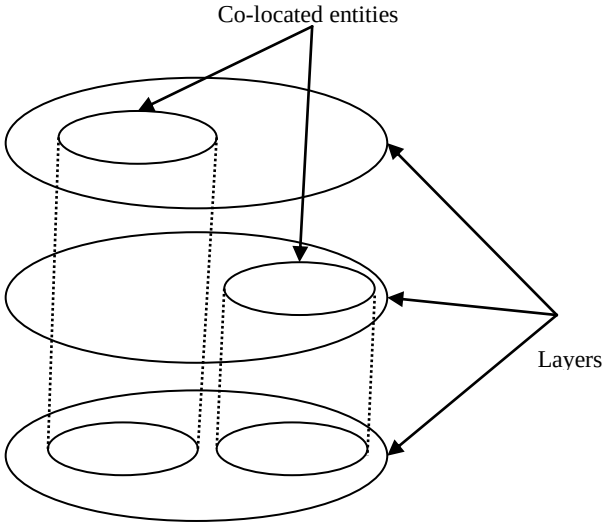


Fig. 11. Relative locations in layered mereotopology

The connection relation in layered mereotopology departs from its axiomatization in non-layered mereotopologies. It is reflexive, antisymmetric, while monotonicity does not apply. It respects the following axioms:

1. If a thing is part of another thing, then everything connected to the first thing is connected to the second one;
2. If two things are connected, then they are parts of the same layer;
3. If two things are connected, their regions are also connected;
4. If two things belong to the same layer and their regions are connected, then they are connected.

Layered mereotopologies have capacities to deal with all types of coincident but non-overlapping entities. The categories represented by layered mereotopology may include not only the material objects and their regions, but also other types of entities such as: qualities, processes, collectivities, systems, wholes. This is crucial in a those of domains that exhibits far reaching complexities. Manufacturing and logistics provide many examples of that kind.

Following the discussion of layering a twinning topic can be addressed, i.e. granularity. While mereotopologies can be productively used for qualitative spatial reasoning, they face difficulty when dealing with those kinds of reasoning which involve the factor of granularity, i.e. when it is needed to consider different resolutions. This is because such a theory would prerequisite means of talking about objects at different resolutions, without at the same time talking about their parts. However in mereology, if an object falls within some range over which it is quantified, then do so all its parts. This issue is due to transitivity of parthood relation. It can be solved by adding to the mereotopological approach a theory of granular partitions (Bittner and Smith, 2003a).

Many examples of granularity are provided by the manufacturing and logistics domain, like e.g. with regard to: time, demand, capacities, locations etc. In each of these cases a certain grid of labeled cells could be employed, and the certain objects could be viewed as located in some exact cells. Such a grid of labeled cells is an example of a granular partition. A granular partition may be approximately conceived as a mereological sum of its constituent cells (Bittner and Smith, 2003a). Partition maybe eventually arranged in a certain order, e.g. it can be timed (Fig.12).

Granular partitions normally respect the covering axiom and the closure axiom (Stevens, 2011). Examples of granulated partitions are the enumerated classes. Actually granular partitions are involved within various activities, like: planning, scheduling, mapping, sorting, listing, naming, counting et al.

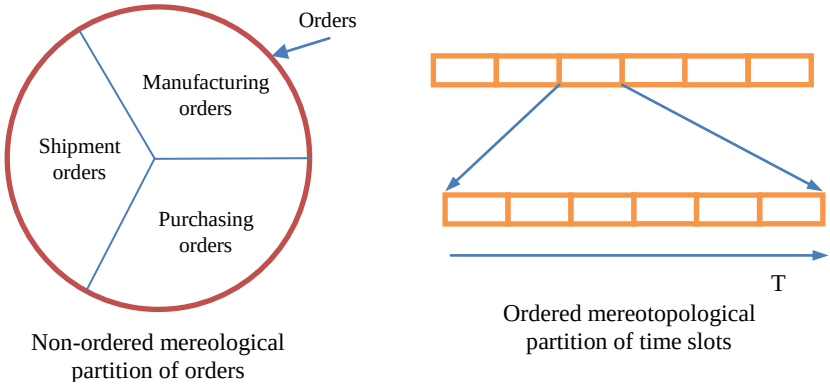


Fig. 12. Illustration of partitioning-based granularities

The subcell relation holds between two cells independently of whether there are any objects located in them. It is reflexive, antisymmetric, and transitive. The cells within a granular partition may however exhibit properties which the singletons of set theory lack. This is because, where a singleton is defined in the obvious way in terms of its member, each cell of a granular partition is defined by its label, and this means independently of any object which might fall within it. To get sense of this question one need to distinguish the mereological sum of two cells from what could be named their partition-theoretic sum. The last one can be defined just as a result of taking two cells together in an abstract way and treating the result as a whole.

Mereological sum apply to both cells and objects, while partition-theoretic sum applies only to cells. The cells of a partition are what they are independently of whether there are objects located within them. Partitions do not only recognize objects. They are also capable to reflect on the mereological composition of objects, which are recognized through a corresponding mereological structure on the side of their cell array.

Finally, the theoretical foundations of temporal aspects in ontologies should be addressed. It is the more that dynamics complexities provide the major issues for the planning and control in manufacturing and logistics. The common literature approaches focus on spatiotemporal frameworks, to account for the diversity of things in

their relations to time, change, and persistence in an integrated way (Grenon, 2003a). These frameworks extend the existing mereotopologies by adding new or re-interpreted abstractions and developing relevant theories. A simple illustration of such basic commitments, with regard to the connection relation, is presented in Fig.13.

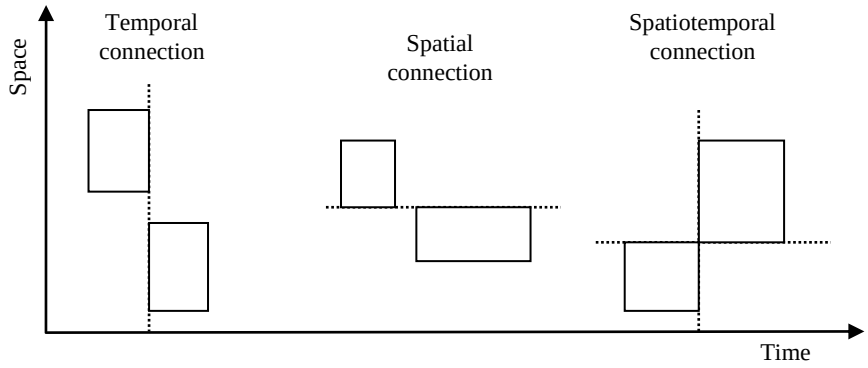


Fig. 13. Temporal, spatial and spatiotemporal connections between spatiotemporal objects

Spatiotemporal mereotopology has to consider the variety of modes of existence and persistence in time to describe both, complex spatiotemporal processes and the enduring entities, that participate therein along performance of processes. Therefore two types of entities are typically distinguished:

- Continuants in time, i.e. enduring entities: they persist by continuing to exist at different times or during periods of time and preserve their identity through change;
- Occurrents, i.e. perduring entities: they are bound in time, i.e. they occur or happen at a time or throughout some period of time, and when persisting they are doing so by having successive temporal parts.

Entities of each type have a specific relation to time, which is an additional locative dimension for the latter, not for the former. Interestingly the literature displays a strong tendency to gloss over the distinction between processes and events.

Following the above two genuine modes of being in and persisting through time are commonly recognized, which are unveiled by distinct perspectives upon reality (Grenon and Smith, 2004). The first one addresses the occurrents and is referred as the SNAP (from snapshot). The other one, referred as the SPAN (from span of time), focuses on the continuants. Basically each SNAP ontology is indexed by a single time instant, while each SPAN ontology is indexed by a single time interval. Using the SNAP framework different dynamic conceptualizations can be supported.

Spatiotemporal frameworks extend the mereotopological perspective on the concept of location. It is possible to distinguish two forms of abstraction with respect to which sound formalisms might be elaborated:

i. spatiotemporal locations as projections of processes at an instant or over some time period;
ii. temporal locations as projections of processes upon the time dimension.

An important format in temporal knowledge representation are the symbolic interval time series. In particular, numerical time series can be converted to symbolic interval time series by segmentation, discretization or clustering. Such approaches are natural for the manufacturing and logistics domain, which with the exception of real-time control normally discretizes the time. Examples of the interval time series are calendars and time buckets.

The most common way for temporal relationships defining are the time intervals operators proposed by Allen (1983). The 13 interval relations of Allen are 'equals', and the following six relations together with their corresponding inverses:

- during$<t_1, t_2>$: time interval t_1 is fully contained within t_2;
- starts$<t_1, t_2>$: time interval t_1 has the same beginning as t_2, but ends before t_2 ends;
- finishes$<t_1, t_2>$: time interval t_1 shares the same end as t_2, but begins after t_2 begins;
- before$<t_1, t_2>$: time interval t_1 is before t_2, and they do not overlap in any way;
- overlap$<t_1, t_2>$: interval t_1 starts before t_2, and they overlap;
- meets$<t_1, t_2>$: interval t_1 is before interval t_2, but there is no interval between them, i.e., t_1 ends where t_2 starts.

The operators can describe any relative position of two intervals. They are of common use for formulation of temporal rules involving intervals. Alternatively, the intervals can be obtained directly from other temporal data, or from association rules over time.

Interval relations can be applied to mine for patterns. However, with this regard Mörchen (2006) exposed that small differences in boundaries lead to different patterns in similar situations. Consequently Allen's temporal relations, which at the first glance look obvious and elegant, are ambiguous in nature, not scalable, and not robust.

Instead Mörchen proposed as an alternative the Time Series Knowledge Representation (TSKR), which is a logic based approach to describe temporal constraints with multiple granularities related to event occurrences. The TSKR expresses the temporal concepts of coincidence, allows partial order, and considers subintervals in patterns. Although the TSKR does not exhibit significant advantages for the upper levels of planning and control, and is not suitable for real-time control of physical devices, it may be useful for the shop floor level control of operations. Herein the requirements of time-rigidness normally exhibit moderate level.

The SNAP-SPAN framework is exploited by another approach, which applies the temporal graphs aiming to merge them with specifications of the W3C (Gutierrez et al., 2007). The dynamics of the phenomena is represented by a series of graphs that separately describe successive moments or intervals of time when the phenomena exists. The labeling of changing objects is considered as the mechanism for adding the time dimension to graphs. The changes are represented in graphs by arcs. They are attributed by time intervals, or alternatively by time points of their validity. The temporal entailment is also incorporated into the framework.

Some other aspects of spatiotemporality are also discussed in the literature, like: time-related granularity (Bittner and Smith, 2003b); space-time continuity (Cohn and Hazarika, 2001); indexing items by time (Grenon, 2003a); transition rules for spatio-temporal histories as an analogue to scenarios (Cohn and Hazarika, 2001).

Grenon and Smith (2004) distinguish changes and processes. Following Ingarden (1964), they interpret processes as a temporally extended occurrents. This departs from the common understanding of this term in manufacturing and logistics. Grenon and Smith recognize changes are as the entities in their own right, including:

i. Qualitative changes: they refer to transformation of characteristics, while identity of items is preserved:
ii. Spatial and locational changes;
iii. Substantial changes, i.e. with regard to the creation or termination of items; e.g.: initiation, termination, splitting, assembly, composition, et al.

Although temporality is a common topic, most papers focus on particular aspects of the phenomena, i.e. existence or persistence of items along the time dimension or spatiotemporality. Despite that e.g. Bhatt (2012) claims in his recent publication that "areas of … reasoning about action and change are mature and established tools, formalisms and languages from therein are general enough to be applied to the case of dynamic spatial systems, where relational spatial models undergo change as a result of interaction (i.e., actions and events) occurring within the system or environment being modeled". Unfortunately the existing meta-ontologies and formalisms (Herre et al., 2010), like the situation calculus, event calculus, process calculus, or fluent calculus, do not sufficiently meet the needs of planning and control in manufacturing and logistics. There are several reasons for that, but the following two are most important:

A. The literature centers on the relation of connection with regard to the time dimension. The concepts of temporality and spatiotemporality are both built around this mereotopological abstraction. However in manufacturing and logistics the primary concern is not how actions connect on the time scale, but how they relate each to other in terms of qualitative temporal relations, like precedence, sequencing, concurrency, synchronization, mutual exclusion, succession and so on. These relations normally exhibit limited time-rigidness, but are rather concerned with some non-timed interdependencies, like those between actions and resources, as well as concerning availability of resources, contextual conditionings, and similar factors. The time-rigidness is directly important only when some actions have due times assigned, or when the operations run in highly automated environments, i.e. when the real-time controls takes place.
B. The major concern of planning and controlling manufacturing and logistics operations is how to harmonize them, or in other words, how to sustain the actions running, while considering various complexities on the way. Therefore at the end of the day it appears that the prior issue herein is, how to control the flows of actions, while the concern about ongoing spatiotemporal connections expressed in a mereotopological way may be eventually a secondary issue.

The above suggested dichotomy, which exists between the common ontological commitments regarding temporality, and the requirements of the domain of interest, is interestingly visible in the argumentation provided by Galton and Mizoguchi (2009). The authors recognize, like many others do, that an exclusive focus on objects (or oppositely, processes), is inappropriate due to the basic mutual interdependency between themselves. However, when it comes to details, it is not possible to agree with the presented argumentation about properties of the two types of abstractions[2]. In particular, the following objections have to be stated:

- It is not always that processes exhibit non-discrete nature (see chapter 1), like it is suggested by Galton and Mizoguchi (2009). The processes and their elements are lasting individuals, but this is a different property than discreteness.
- It is not always that objects are non-dissective. E.g. a part of manufacturing cell may also exhibit the properties of a manufacturing cell.
- The interpretation of process dissectivity (homogeneity) suggested by Galton and Mizoguchi does not apply in the manufacturing and logistics domain, as sometimes a portion of given process may be of different type, than its ancestor.
- Similarly with the property of indefinite extension. Galton and Mizoguchi presume any process as going on throughout an interval of time, and its subintervals, or at each instant during any interval throughout which it is going on. Again, this is not always the case in discrete manufacturing.

The above example is one more illustration given in this chapter, of how important is to adjust the grounding abstractions to a particular domain of interest.

An adequate formal framework for spatiotemporal knowledge representation and reasoning should have to elicit both, the enduring and perduring entities, with a special respect given to their peculiarities and couplings. It should have resources to define boundaries and interdependencies of perdurants, and thus bounded or connected processes, their components, etc. With such capacities it would become possible to provide precise characterizations of various spatiotemporal categories, and to support the knowledge-aided activities. For what concerns planning and control in manufacturing and logistics, seven distinctive characteristics of the domain can be named:

i. The major concern of planning and controlling operations is to assure their sustainable run. Therefore the core ontological focus should be about the representation of anticipating and triggering different operational activities and processes.
ii. It often happens that despite the evidently spatiotemporal nature of the phenomena, the time dimension does not need to be directly addressed. A simple example is presented in Fig.14. The subject of planning herein, which goes on in a run-time mode, is the logistical operation of semi-automated pick-to-light. The single task (i.e. the ad hoc composed operation) for the operator is to collect a number of items

[2] To note, Antony Galton is internationally recognized scholar. His views cannot be questioned as mistaken. They are just not valid in the domain of manufacturing and logistics operations. Unfortunately Galton and Mizoguchi do not recognize, like many other authors, that their propositions are not as generic, as they seem to be to their proponents.

in the stockroom. The flashing lights guide him and indicate where to collect the next item, and which one should it be. The focus of ontology-aided planning is the productivity aspect, i.e. reduction of walking distance. With this regard the layout of stockroom has to be considered. It exhibits a two-dimensional spatial discrete structure, but along the planning of operator's route it should be rather considered as a linear one, with stock buckets attached to the possible stops (Fig.14).

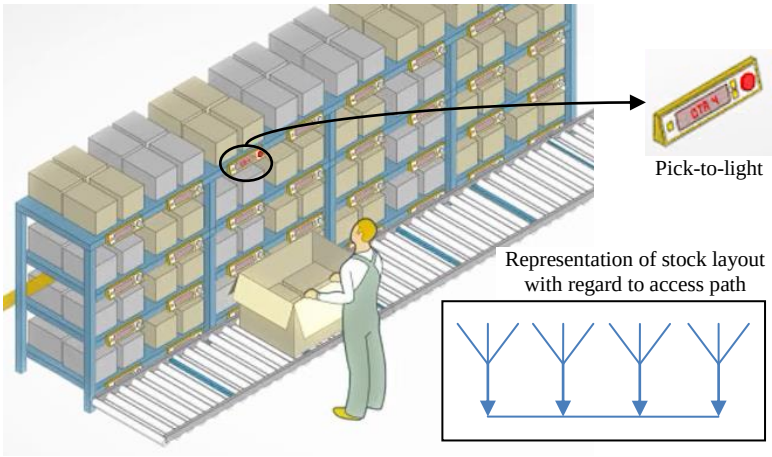


Fig. 14. Representation for run-time planning of pick-to-light operation

iii. The concern about the time flow may be more or less rigid, depending on the specific circumstances. In the example pictured at Fig.14 all entities subject of planning, except the operator's eligible route, are perdurants. Despite that, the timing aspect is not directly considered at all, starting from the requirements for items, which are rather prioritized than timed. Timing is just not required. Of course, as a consequence of performed operations all entities could be timed ex post, when it would be needed.

iv. The variety of types of perduring entities is very reach, in comparison to other ontological domains discussed in the literature. Furthermore, many concepts normally considered as endurants, exhibit perdurant qualities herein. The best example is resources, which are subjected to frequent transformations along the processes. They are frequently reconfigured, i.e. (de)composed into new spatiotemporal entities of requested structures at available locations, according to the changing needs.

v. The peculiar characteristic of the manufacturing and logistics is the extremal reach of dynamic interdependencies, which manifest themselves along the planning, controlling, and the course of operations. Hence, all of them have to be considered on the way, and built into the ontology which is expected to aid these activities.

vi. Typical functionalities of the core ontology require involvement of many categories, both mereological and spatiotemporal, like e.g.: calendars, demands, plans and schedules, profiles of resources, and so on. Each category is normally supported by a separate algebra and/or mereotopology. Therefore the question arises about matchings between categories and mappings between entities. E.g. in the above ex-

ample two different spatiotemporal categories, both representing the layout of stockroom and the stock, have to be matched by a relevant transforming functor. To provide a transparent picture of this issue, let us refer to a simple example of product structures and schedules. The product structure is handled by the 'used-for' relation ⊑. Both, the ⊑ relation and the items, are featured with numerical properties, which are 'material standard' and 'lead time' respectively. The push flows based planning and control is assumed. Using the above assumptions, minimum algebras for the categories of products and schedules can be defined and axiomatized. The matching between them can be expressed by the **PS** and **PS** functors, which provide the means to map objects between categories (Fig.15).

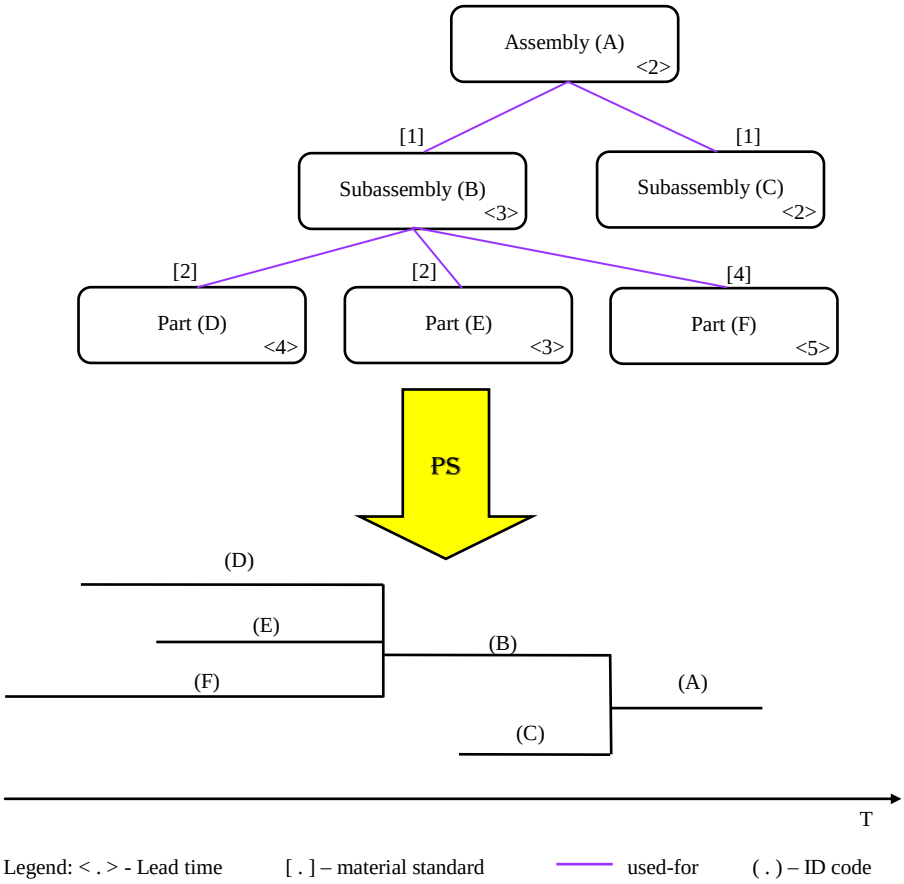


Fig. 15. The category of products is transformed by the functor **PS** to the category of schedules

To ease the functors are expressed in an algebraic form in the below definitions, which also address some elements of the algebra for the category of scheduled requirements:

$$\mathbf{S} = \mathbf{PS}\{\ \mathbf{P}\ \}$$

$$\mathbf{P} = \underline{\mathbf{PS}}\{\ \mathbf{S}\ \}$$

$$\underline{\mathbf{PS}} = \mathbf{PS}^{-1}$$

where:

$\mathbf{P}$ - category of products (i.e. built upon the mereological sums),

$\mathbf{S}$ - category of schedules (i.e. built upon the spatiotemporal mereotopology).

The $\mathbf{PS}$ functor is defined as follows:

$$\mathbf{PS} := \mathbf{PS}\{\mathbf{EX}\{\ .\ \} \otimes \mathbf{OF}\{\ .\ \} \otimes \mathbf{NR}\{\ .\ \} \otimes \mathbf{LS}\{\ .\ \} \mid \forall i \forall j \exists \{i \sqsubseteq j\}\}$$

where:

$\otimes$ - superposition of operators

$\mathbf{EX}\{\ .\ \}$ – demand exploding operator,

$$\mathbf{EX}\{\ .\ \} := \forall i \forall j \exists \{i \sqsubseteq j\}\ \mathbf{EX}\{j, i\} = \oplus\!\!\rightarrow \{\underline{\mathbf{RQ}}_{\tau}(i) = \mathbf{NR}_{\tau}(j) \times [i, j],\ \tau > 0\},$$

where:

[i, j] – material standard of item i used for item j, subject to $\boldsymbol{\varepsilon}$ relation,

$\oplus\!\!\rightarrow$ - schedule composing operator,

$\underline{\mathbf{RQ}}_{\tau}(i)$ – non-offset gross requirements (demand) for item i at time unit τ,

$\mathbf{NR}_{\tau}(i)$ – net requirements (demand) for item i at time unit τ,

and

$\mathbf{OF}\{\ .\ \}$ – demand offsetting operator,

$$\mathbf{OF}\{\ .\ \} := \forall i\ \mathbf{OF}\{\underline{\mathbf{RQ}}(i), \mathbf{RQ}(i)\} = \oplus\!\!\rightarrow \{\mathbf{RQ}_{\tau}(i) = \underline{\mathbf{RQ}}_{\tau+<i>}(i),\ \tau > 0\},$$

<i> - lead time of item i,

$\mathbf{RQ}_{\tau}(i)$ – gross requirements (demand) for item i at time unit τ:

$\mathbf{NR}\{\ .\ \}$ – requirements (demand) netting operator,

$$\mathbf{NR}\{\ .\ \} := \mathbf{NR}\{\ .\ \} = \forall i\ \forall \tau > 1\ \oplus\!\!\rightarrow \{\mathbf{NR}_{\tau}(i) := |\mathbf{OH}_{\tau}(i)|\ \mid \mathbf{OH}_{\tau}(i) < 0\}$$

where:

$\mathbf{OH}_{\tau}(i)$ – on-hand operator for item i at time unit τ,

and

$\mathbf{LS}\{\ .\ \}$ – lot-sizing operator,

$$\mathbf{LS}\{\ .\ \} := \forall i\ \forall \tau > 1\ \mathbf{LS}\{\mathbf{NR}_{\tau}(i)\} = \oplus\!\!\rightarrow \{\ L_{ri}^{j}\ \}$$

where:

L_{ri}^{j} - j-th lot (proposed order) of item i for due time unit i,

and additionally:

$$\mathbf{RQ}_{\tau}(i) := \sum_{j \neq i} \mathbf{NR}_{\tau+<j>}(j) \times [j] \mid \forall\ \{i\ \boldsymbol{\varepsilon}\ j\}$$

$$\mathbf{OH}_{\tau}(i) := \mathbf{OH}_{\tau-1}(i) + \mathbf{SR}_{\tau}(i) - \mathbf{RQ}_{\tau}(i)$$

where:

$\mathbf{SR}_\tau(i)$ – scheduled receipts for item i at time unit τ,

where:

$\mathbf{SR}\{\ .\ \}$ – scheduling receipts operator,

$$\mathbf{SR}\{\ .\ \} := \forall i\ \forall \tau>1\ \mathbf{SR}\{\oplus\{\ L_{ri}^{j}\ \}\} = \oplus\{\ \overleftarrow{L}_{ri}^{j}\ \}$$

where:

$\overleftarrow{L}_{ri}^{j}$ - j-th planned order (scheduled receipt) for item i at time unit τ.

It must be underlined that inversibility of the functor **PS** depends on how the ontology is constructed. It can protect morphism of two categories or not[3]. With this regard the distortion of demand may be tamed or enhanced. On the other side, the robustness of ontology-aided solution is dependent on how the timing is robust or not, as was e.g. illustrated in the aforementioned publication of Mörchen (2006). However, the latter aspect is a separate issue. A well designed ontology should protect, if possible, isomorphism between its distinctive categories, as well as their equivalence. Furthermore, in such a case, and in particular when the functor between categories is inversible, the ontology can be simplified and diminished, for the advantage of its efficiency and robustness. This is a very generic issue.

vii. Ubiquitous layering and granularity provides another specific dimension to the complexity. This aspects may refer to various categories, like: resources, products, processes, layouts, calendars, demand, requirements, plans, schedules etc.

The above listed specifics of the domain set particular and challenging requirements, which have to be met when designing its meta- and core-ontology.

Following the recommendations of Gruber (1993), the concise but comprehensive review of meta-mathematical foundations for developing ontologies presented in this section, can be resumed by a set of ontological commitments. A similar and extensive proposal was provided by Hahmann and Grüninger (2012), however in reference to the mereotopology. The ontological choices discussed in this section are summarized in Table 1. Each aspect is presented as a partitioning of the available choices. The set of commitments is tailored to meet the specific requirements of the analyzed domain.

The discussed meta-mathematical theories are by their nature very high-level, and to some extent selective. While much research work on ontologies has been primarily theoretical, recently the focus started to shift towards domain-specific extensions and adjustments, to better suit practical needs. This aspect is likely to guide the future direction of the field. The integration of more comprehensive and pragmatic abstractions and formalizations, especially with regard to the dynamic complexities, provides additional challenge.

[3] The existing ERP systems do not use ontologies. They typically apply standard data structure and algebraic solutions for the MRP computations that do not preserve morphism between the product structures, i.e. Bills of Materials, and the material requirement structures. That is why they amplify distortion of demand, turbulences in supply chains, and why they exhibit the so called nervousness (Heisig, 2002). The above example meets the ideas of MRP.

Table 1. Ontological commitments and possible choices

Ontological aspect	Options
Founding theories	Mereology Ontology Mereotopology Prototethic
Sums	Ontological sum (distributive classes) Mereological sum Mereotopological sum
Elements/parts	Mereological parthood (transitive) Ontological elementhood (non-transitive)
Mereotopological aspects	Spatial connection Spatial coincidence Regions Region-free Overlapping Underlapping Containment Layering partitions Granular partitions (sub-celling)
Temporal aspects	Temporal connection Endurants Perdurants Indexing by time instants Indexing by time intervals Processes Events States Transition rules Changes: qualitative vs. substantial Graph-represented changes Timed granularity
Spatiotemporal aspects	Spatiotemporal connection Spatiotemporal locations Spatiotemporal regions Spatial / location changes Spatiotemporal histories

The theories of ontologies are not directly interesting for those who design the domain ontologies. They are primarily exploited by the community developing the means of ontology engineering. The design of ontologies is much affected by the technological aspects, i.e. with regard to the limits of capacities offered by languages and other resources of ontology engineering. Nevertheless, a good understanding of grounding abstractions and formal foundations is crucial for effective and efficient development of ontologies. Otherwise they might be subjected to surprising shortcomings and faults.

A spectacular example of such a case is the ISO 15926 Standard ‘Lifecycle Integration of Process Plant Data Including Oil and Gas Production Facilities’, which is

an example of manufacturing system ontology, and has been proposed for general use as a core ontology for the domain. When examined, it appears to be marked by a series of astonishing defects which justify a question if the standard is actually an ontology (Smith, 2006). The failures of ISO 15926 seem to be caused by an incorrigible understanding of the grounding ontological concepts by its authors.

The above argumentation confirms that developing a well-founded ontology for manufacturing and logistics operations is not easy. Rigorous formal framework has to be supported by a carefully designed methodology, while good understanding of theoretical foundations by ontology developers cannot be overestimated. This issue is especially important for the domain of manufacturing and logistics operations. As it was argued in chapter 2, it is still in an early stage of its development, and as it was argued in this section it provides particular challenges. This issue will be continued in next sections, as well as in section 3 of chapter 8, and in chapter 10.

3 Ontology engineering

As it was argued in the introduction, ontologies can be used as core conceptualizations for different applications applied in various domains. This means that ontology is understood as a formal representation of a shared and common knowledge about a particular domain (Gruber, 1993). The concept of ontology was adopted in 1980s by the domains of artificial intelligence and computer science from meta-mathematics and mathematical logic. Then in 1990s, after publication of the seminal work of Gruber (1993), it was widely disseminated. Primarily ontologies in the ICT domain are considered as means to communicate and share knowledge between humans and application systems, in a transparent and consistent way.

Following the argumentation of Davis et al. (1993) ontologies can be assessed as knowledge representation frameworks according to the following five perspectives:

1. As a surrogate of knowledge.
2. As a medium for information processing.
3. As a set of ontological commitments.
4. As a fragmentary theory for reasoning, which includes sanctioned and recommended inferences and related conceptions for representations.
5. As a medium for expressions, including the human-oriented one.

Ontology engineering provides principles and methods for activities and processes that support creation, exploitation, management, maintenance and reuse of ontologies. The common tools of ontology engineering are methodologies, ontology languages, ontology editors, ontology visualizers, reasoners (e.g. inference engines, theorem provers, classifiers et al.), metrics etc.

A particular property of ontologies, which enables a clear distinction between the ontologies and the system models is the so called open-world assumption, which states that anything not explicitly expressed in a conceptualization is unknown. Hence ontology is understood as a partial description and accepts under-specification as a mean of abstraction. Oppositely, system models assume that what has not been speci-

fied is either implicitly disallowed, or implicitly allowed, which is the so called closed-world assumption.

Deviating from its original philosophical meaning, in the context of ontology engineering the term ontology is understood as a formal explicit specification of shared conceptualization (Gruber, 1995). Such a rigid interpretation may confuse, as it is not aligned with the preceding argumentation. Nevertheless it is understandable, as ontologies are expected to be encapsulated into the software models, or directly into the applications (Hoehndorf et al., 2009). In such a context they have to describe a given domain as completely as possible, with regard to the rigid requirements of software engineering.

In the management science models are commonly understood as partial and explicit representations of a given reality as it is seen by those people, who wish to use them to understand, change, manage, and control that part of reality (Seidewitz, 2003). This definition exposes another possible distinction between the models and the ontologies. Model is an abstraction from reality developed as its specification, i.e. it reflects structure and behavior of some part of reality. Model is developed for specific uses. According to Seidewitz models are prescriptive, while ontologies are descriptive.

Principally ontologies can be used for various purposes, including run-time uses by different applications. Hence ontologies can be reusable. Furthermore, ontologies respect accept under-specification within their abstractions, not like the models. They can also incorporate some particular modes of under-specification, like e.g. in terms of the grey conceptualization and modelling (Lin and Liu, 2006).

A summary of the major distinctions between ontologies and models is provided in Table 2.

Table 2. Dozen distinctions between ontologies and system models

Ontology	**System model**
open-world assumption and under-specification accepted	assumption of closed-world and rigid specification
descriptive	prescriptive
formal	not necessarily formal
no focus on realization	focus on realization
enable knowledge exploitation	knowledge exploitation impossible
possible run-time use	run-time use impossible
reusable	limited reusability, if at all
can support reasoning	cannot support reasoning
can be manipulated by machines	cannot be manipulated by machines
extendable and mergeable	non-extendable
reconfigurable	reconfiguration requires recompilation
accept vagueness	reject vagueness, unique names needed

Recent progress in software engineering is centered on the development and use of models. Models are specified using formal languages, e.g. UML, and then used directly to generate codes. Software models require a method for both, making the domain knowledge explicit, and for integrating it with the conceptual model of the application (Bézivin, 2005).

The contemporary approaches to software development are limited with the above regard, as they are not able to separate conceptual and domain models. Hence, software cannot be ported between different domains without altering its conceptual model, and consequently its whole model. As it was explained before, these shortcomings can be avoided by using ontologies. A relevant approach for this purpose is presented below (Fig.16).

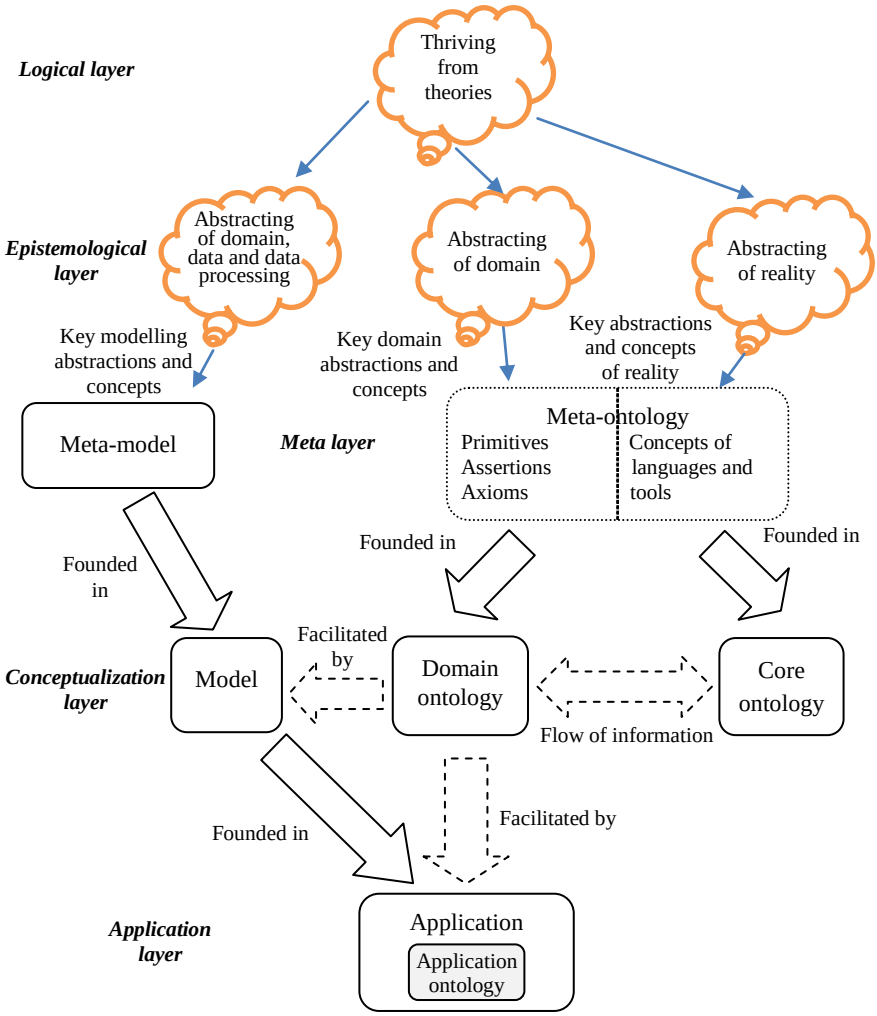


Fig. 16. Founding and integrating models, ontologies and applications along their development

Aiming at synergy of ontologies and models driven applications development, the presented integration scheme can be defined. It makes distinction between the meta-ontologies, core ontologies, domain ontologies and application ontologies.

The meta-ontology is a result of the epistemic understanding, which is normally too much extent affected by the logical abstractions of particular theories, which in our case are: mereology, ontology, prototheticc, and mereotopology. The cognitive process results in provision of basic abstractions and conceptualizations. These may be general, and in such case they are embedded into the ontology languages and tools. Alternatively, they can be domain focused primitives and other basic abstractions, like class and property axioms and assertions, class expressions, et al. Therefore a domain meta-ontology is composed of two parts, which are general and universal.

The core ontology extends the meta-ontology by all domain specific universals, like classes, properties, restrictions, quantifications, some individuals, et al. An example could be the ontology for manufacturing and logistics.

The core ontology is still a general purpose development, not like the domain ontology, which focuses on some range of application domains, i.e. by a further extension and refinement of the core ontology. An example could be the ontology for planning and control of manufacturing and logistics operations.

The application ontology addresses some range of applications developed for the problem domain, i.e. the scope of problems that an application (software) is intended to service. The application ontology is normally used in a distributed run-time mode.

The proposed scheme enables to overcome the fundamental divergence, which still exists between current developments in the areas of ontology engineering and software engineering. It also provides a grounding idea for integration of various domains, their activities, and supporting applications, which have to be operated in a networked environment. Particularly, the domain ontologies may provide replaceable components within application architectures.

The presented scheme exploits ontological foundations of conceptual modeling, as well as other advances of semantic technologies. It is technically viable, e.g.: foundation relations between different items may be serviced by mappings, ontological translations or reductions; inconsistencies that may arise when ontologies are being merged, can be automatically detected and eliminated; etc.

The structure suggested for integrated use of ontologies and models along application development can be imitated when the applications run. Different applications can share and exchange ontologies, eventually in the run-time mode. This way an intensive cross-layer, cross-functional, cross-domain and cross-process integration is made possible.

The exchange of knowledge within distributed ontologies may go in parallel with the data exchange that is normally supported by integrated database management technology. However, as it was argued above, from the functional perspective, ontologies have very different functional and operational role, than just the acquisition and exchange of data or information.

Current approaches to software development emphasize modularity and reusability of applications. Facilitation of such features requires both, an understanding of the

tasks to be performed by the software, and the knowledge about given domain. Such qualities can be provided by ontologies.

Ontology may be externalized in many ways, including provision of external ontology services. This aspect exposes a particular side issue: whether ontologies should be standardized for some domains. Alternatively bottom-up development may lead to incoherence of domain ontologies. Possibly, in order to be general enough, ontologies should be eventually negotiated between their stakeholders, and be confirmed by competent institutions, like: standardization committees, professional associations and scientific organizations. This sets a question about: intelligibility, internal coherence and consistency, expressive adequacy, completeness, vagueness, reasoning efficiency, uncertainties, vastness, and finally failures of shared ontologies, like it was illustrated by a spectacular example in the preceding section. Deceits should be also taken into consideration.

The use of ontologies by information systems can be compared to the database aided information processing (Martinez Cruz et al., 2012). With this regard two basic parts of any ontology can be distinguished:

i. axioms, which describe the structure of model upon which an application is based;
ii. set of the facts describing some particular situation, current circumstances, etc.

Ontology axioms behave like implications (inference rules) and entail implicit information. They are analogous to a database schema, which describes structure of data and constraints on data. Ontology facts are analogs to a database data, as they instantiate schema and are consistent with its constraints. However, with regard to querying interesting differences between ontologies and relational databases exist. Database schema behaves as constraints on structure of data and define legal database states, but plays no role along query answering, which amounts to model checking and can be very efficiently implemented. Oppositely, ontology axioms play a crucial role along querying. Query answering may incorporate implicitly derived facts. It amounts to theorem proving and can support both, conceptual as well as extensional queries. Unfortunately, the price for these welcome features is high complexity. Therefore ontologies should be used when:

i. structure of information is extensive and/or complex;
ii. structure of information is used along querying;
iii. modeling of complex structures/ activities is crucial;
iv. complete information is unavailable;
v. inference is requested along querying;
vi. reasoning is needed to structure and validate information.

Contemporary ontologies follow the theoretical foundations set by Leśniewski and share many structural similarities, regardless of the language in which they are defined. Basically ontologies include:

- Classes/concepts: sets, collections, types of objects. They may contain individuals, other classes, or a combination of both.
- Relations: ways in which classes and individuals relate to each other.

- Individuals: instances, objects et al.
- Attributes: properties, features, parameters, aspects and other characteristics of classes, relations and objects.
- Events: result in changes of attributes or relations.
- Axioms: assertions (including rules) in a logical form which comprise the overall theory that the ontology describes in its domain of application.
- Restrictions: formally stated descriptions of what must be true in order for some assertion to be accepted as input.
- Function terms: complex structures formed from certain relations that can be used in place of an individual term in a statement.
- Rules: statements in the basic form of "if-then" sentence, to describe the logical inferences that can be drawn from an assertion in a particular form.

Contemporary ontologies refer to different knowledge representations and use a variety of languages which rely on distinct grounding theories, like: Frames, Rules, First Order Logic (FOL), Modal Logics, etc. Majority of existing ontology languages are declarative and either use Frames, or are based on the FOL (Maniraj and Sivakumar, 2010). The ultimate knowledge representation benchmark, in terms of compactness and expressive power of its formalism, is the FOL. However, it has two drawbacks: difficulty to use, and impracticality of implementation, especially with regard to performance of applications. Thus, some subsets of FOL are commonly used. E.g. the Semantic Web initiative widely refers to the Description Logic (DL).

Languages based on the automatic classification are expected to support the new layer of semantics on top of Internet. Instead of searching via text strings, as it is now in Internet, Semantic Web will make possible logical querying and find pages that map to those queries. The automated reasoning component use classifiers, which focus on the subsumption relations in a knowledge base rather than rules. A classifier can infer new classes and dynamically change the ontology as new information becomes available. This capability is ideal for the continuously changing information in the Internet, while not so obviously to the requirements of planning and controlling manufacturing and logistics operations (Vrba et. al., 2011).

The OWL language recommended by W3C is an extension to the Resource Description Framework (RDF), which is a simply language for description of Web resources (i.e. data models) and relations among them (Hitzler and Janowicz, 2011). OWL extends its vocabulary used to describe classes and properties. Then by using rules and SPARQL querying language provision of web services is made possible.

Many ontology languages, like the OWL, are equipped with syntax and semantics limited to the representation of binary relations. This provides a particular difficulty when dealing with spatiotemporal complexities, as they claim for at least ternary predicates to represent temporal relations in a most convenient way. Although in most cases it is possible to overcome this issue by using binary predicates (Grenon, 2006), it is at a price of highly deteriorated intelligibility. Possibly in reference to some particular complexities a reasonable expression might be not available, if at all.

Among the ontologies a special attention should be given to the meta-ontologies, also called upper ontologies, top-level ontologies or foundation ontologies. They are

environments for development of a core domain ontology, which preserve interoperability. Among the existing ones BFO (Basic Formal Ontology) (Grenon, 2003b), Cyc (Matuszek et al., 2006), DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) (Grenon, 2003b), GFO (General Formal Ontology) (Herre, 2010) and SUMO (Suggested Upper Merged Ontology) (Niles and Pease, 2001) are commonly regarded as most comprehensive and advanced. What concerns the spatiotemporality, DOLCE treats perdurants following Vendler (1967), who proposed a four-fold distinction of achievements, accomplishments, states, and processes. In DOLCE, only the first two categories count as events, which are thereby separated from processes. In BFO occurrents are divided into spatiotemporal regions, temporal regions, and 'processual entities'. Within the latter category the most pertinent distinction is between processes and their boundaries. Process boundaries correspond to boundary events, while all other events are recognized as processes. BFO treats states as continuants, which is exceptional. E.g. DOLCE recognizes states and processes as 'static' perdurants. Like BFO, other ontologies such as SUMO and CYC downplay or ignore the distinction between processes and events. All above mentioned meta-ontologies are supported by sound theories. Some of them, especially the Cyc, are tooled with a full set of sound technologies, and have found many advanced implementations. Unfortunately direct utilization of meta-ontologies is sometimes limited because they are often built upon very expressive languages without taking care of efficiency.

The development and design of ontologies raises a question about methodologies to support these activities. Not to forget, exploitation and maintenance of ontologies dilate the matter. The before mentioned ISO 15926 case provides a spectacular warning against underestimation of this issue. To mention, as ontology-aided applications are qualitatively different than the information systems, the existing methodologies for development and design of information systems cannot be directly adopted.

Obviously any methodology to develop or design ontology should be conditioned by specific factors, i.e. with regard to domain and potential scope of implementations of a given ontology. The volume of literature addressing this topic is slim. Despite of that manifold approaches are applied, ranging from high level ones, to very technical and detailed. The existing literature perspectives onto ontology development and design can be categorized into five items, which are as follows:

- Epistemological perspective (Grenon and Smith, 2004).
- Characterization perspective - ontology as a semantic representation of an acting system (Gorbatov, 1979).
- Ontology as a specification mechanism subject to design (Gruber, 1995).
- Ontology as a knowledge representation technology (Brachmann and Levesque, 1985).
- Implementation perspective (Uschold and King, 1995; Fernandez-Lopez et al., 1997; Noy and McGuinness, 2001; Smith, 2006; Jarrar and Meersman, 2009; Martinez Lastra et al., 2010).

The implementation perspective directly addresses ontology development and design as its concern in practice related issues and it takes engineering approach to the topic.

One of the first methodologies to develop ontology called Knowledge Engineering Methodology (KEM) focused on the creation process (Uschold and King, 1995). It encompasses the following phases:

i. definition of the domain;
ii. conceptualization of the domain;
iii. reuse of existing;
iv. formal specification: definition of taxonomies of concepts, attributes and relations;
v. creation of individuals or instances to populate the ontology;
vi. evaluation and documentation.

The framework proposed by Martinez Lastra et al. (2010) takes a similar approach, but focuses on the factory automation domain.

Not like the above two frameworks, the Methontology covers the whole life-cycle of ontologies (Fernandez-Lopez et al., 1997). It is focused on the development of ontologies from the level of knowledge, i.e. similarly to the traditional cascade process applied in Software Engineering. The Methontology defines four phases:

i. definition of scope and granularity;
ii. conceptualization of the domain;
iii. implementation in a language;
iv. evaluation.

Another framework, i.e. the DOGMA (Development of Ontology Guided Methodology Approach) methodology, considers development of ontologies in a very formal manner. They are primarily viewed as scalable and shared resource of knowledge, as the reuse of knowledge is actually the primary concern of DOGMA. With this regard DOGMA proposes the definition of diverse levels of knowledge abstraction, starting from a top level of most general concepts, which can then let these models to be reused in diverse domains.

The implementation perspective is further explored in chapter 10. The framework by Martinez Lastra et al. (2010) is taken as a springboard therein, as it is most adjacent to the domain of interest and directly addresses the ontology development issues, which is a major focus of this book.

However, while keeping the feet on the ground it is also good to have the thoughts high. Therefore more the general viewpoints should not be forgotten, starting from the epistemological one. Other important insights may be also derived by reflecting on the ontology life-cycle and the needs of different stakeholders. Finally, keeping in mind the contextual and functional specificities of various uses, the ontology should not be viewed as a uniform whole, but rather as a collectivity of components, which exhibit the holonic features.

With the above regards and considering the above referenced perspectives[4], a set of fundamental rules for ontology development and design can be proposed. Some of

[4] The view of ontology as a knowledge representation is not considered, as the major concern herein is how to choose or design a knowledge representation tool, like e.g. meta-ontology. With regard to the discussed issue it is obviously a point of interest, but rather a detail.

them are universal. Few presume that ontologies are used as both, knowledge bases and specifications of dynamic systems, at operational or even run-time mode, to aid such sophisticated activities like decision making or automated control.

In line with the above argumentation two epistemological paradigms have been elaborated, which assume that development of any ontology should be preceded by a deep reflection on its grounding logic and abstractions. It is the more that openness is aimed as a crucial feature. The paradigms should be recognized as fundamental pillars for elaboration of any methodology to develop and design ontologies. They are:

I. **Adequate openness**. It opposes the paradigm of reductionism, which claims that among the plurality of alternative conceptualizations of reality there is one particular, to which the others should be reduced[5]. It is also in opposition to the commonly recognized characterization principle of Gorbatov, and to the Gruber's criterion of the minimal ontological commitment. It must be underlined that the reductionist approach is fundamentally irrelevant for development and design of ontologies, as by their nature openness is among the most advantageous and distinctive qualities.
II. **Multi-perspective approach**. It accepts that alternative but equally legitimate viewpoints on reality may exist, e.g. with regard to the requirements of different stakeholders, or various functionalities, or various theoretical foundations, or the life-cycle viewpoint, and so on.

Furthermore, dozen guiding concerns were elaborated which align both, the welcome performance of the ontology, and the (re-)design aspects. They are as follows:

1. **Composability**. It encompasses decomposability, distributiveness, and abilities to cluster, and compose holarchies of components equipped with the holonic features.
2. **Changeability.** It is the ability to adapt to new requirements in terms of analytical content, coupling etc.
3. **Upgradeability**. It is the ability to adopt a new technology.
4. **Fallibility.** It is ability to redevelopment or redesign when a new meta-ontology or new theoretical fundamentals have to be adopted.
5. **Dependability**. Ontology engineering as highly automated technology can be less robust than some more basic 'tried and tested' technologies.
6. **Coherence**. It applies to the logical consistency of ontology, both internal, and with the existing external knowledge expressed in natural language. The domain of interest should be appropriately covered. It also extends to terminological coherence and moderation, definitions should be objective and context independent.
7. **Diagnosability and maintainability**. It is about how errors and faults can be detected, possibly in a preventive mode, and corrected.
8. **Intelligibility and accuracy**. Ontology should be understandable to all. Reality should be correctly captured, axioms should respect the existing knowledge. This capacity may be supported by visualization tools and well done documentation.
9. **Scalability**. It is the ability to shift to a different level of useful capacity.
10. **Inferencing capacity**.

[5] A good example of reductionist methodology in in the ICT area is the Use Case methodology.

11. **Minimal encoding bias**. The encoding bias is when knowledge representation choices are mostly reasoned by convenience of implementation or notation. In such a case it happens that knowledge-sharing applications are implemented in different representations and using different styles (Gruber, 1995).
12. **Efficiency**.

With respect to the above presented guidelines the Ontology DEvelopment and Design (ODED) methodology was proposed to support conceptualization and engineering of Knowledge-driven systems, which encompasses the whole ontology life-cycle. It recognizes ontology as a representation of shared, distributed and evolving knowledge, which is actively used to aid operations, and which is subjected to repetitive reuse and reengineering along its recurrent life-cycle. Fig.17 illustrates the most important activities defined in the ODED methodology as ten phases.

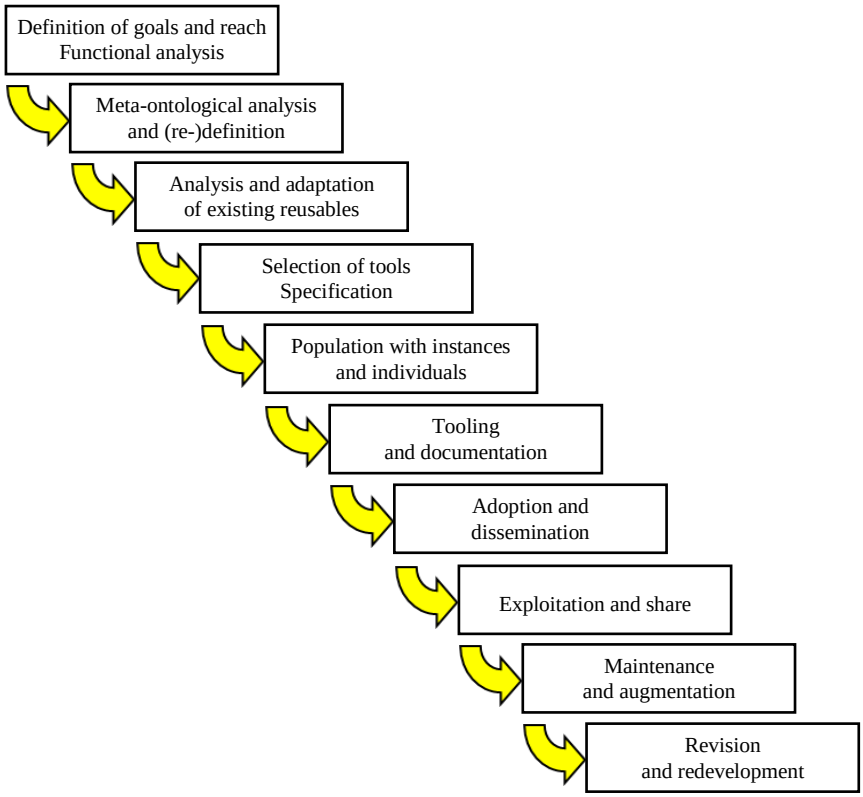


Fig. 17. Waterfall of the major phases in the ODED methodology

To finalize the methodological considerations it must be emphasized, that for obvious reasons a twofold 'global-local' view of manufacturing and logistics ontologies should be respected, i.e. they ought to be considered as: (i) knowledge base aiding systemic solutions; (ii) source of components for embedded appliances.

The above review of ontology engineering in this section provided a number of sound arguments in favor of ontologies, as the means for open knowledge-driven manufacturing and logistics. The major one is the ability to built-in an ontology into embedded appliance or an application, including the run-time mode, hence making the ontology an extendable part of application models. Additional role of ontologies is to support local distributed intelligence. The existing means of ontology engineering, including the most common ontology languages, also OWL, are not well suited to the sophisticated needs of planning and control in manufacturing and logistics. The root cause for that is the limited power of grounding abstractions and founding theories. Meta-ontologies could be a countermeasure against such shortcomings. This issue will be further investigated in the next section. The final doubt is about efficiency. Distributed approach should reduce this deficiency, i.e. by localization and limitation of the issue. Therefore it should be preferred by the beginning developments.

4 Complexity and dynamics of manufacturing and logistical operations

This section investigates dynamic complexities that arise along manufacturing and logistics operations, in particular along the manage processes. The purpose herein is to identify and understand the key types of dependencies and discrepancies aiming their representation within ontologies, thus enabling effectiveness and efficacy of open knowledge-driven operations management.

The two secondary objectives behind the discussion in this section are: (i) to enable verification if the existing capacities of ontology engineering, in terms of grounding abstractions and formalizations, fit well enough to the identified requirements; (ii) to provide a grounding for such a conceptualization of the domain, which could be built into ontologies, while being consistent with the SOA paradigm, hence enabling easy development of effectively running (Web-)services.

The complexity of the analyzed phenomenon can be reflected as granular or layered spatiotemporal, mereological and mereotopological interdependencies and discrepancies. The factors of different complexities in the manufacturing and logistics domain are normally exhibited by some of the following characteristics:

- Diversity: variety of aspects or perspectives which have to be considered; separately, together, or even at once;
- Divergence and latitude: of properties, of choice, etc.;
- Opacity: limited understandability and weak transparency of the domain;
- Discrepancy: inconsistent or conflicting requirements and objectives;
- Spatiotemporal couplings: indispensable fits among various items along the time and space dimensions;
- Mereological or mereotopological couplings: structural ties of items, also with regard to such aspects, like layering and granularity;

- Volatility: variability (i.e. with regard to controllable variations), randomness (i.e. with regard to uncontrollable variations), unpredictability, unforeseenability, idiosyncratic or turbulent behavior.

Next subsections review the identified complexities of planning and controlling manufacturing and logistics operations and discuss them in more details.

Layering of manufacturing and logistics operations and manage processes.
The major focus of manufacturing and logistics operations is to align the demand, resources and fulfillment of demand by manufacturing and logistical processes / operations. The layering of planning and control processes of manufacturing and logistics operations provides a generic factor of complexity for both, operations and manage processes. The typical most extended layered structure of planning and control, which is commonly embraced by the ERP systems, is depicted in Fig.18. The superior operations strategy process provides guidelines to manufacturing and logistics operations, mostly defined in terms of performance measures, i.e. by KPIs. The layers of manage processes in case of push flows are: aggregate planning, master planning, requirements planning, operations planning and operations control.

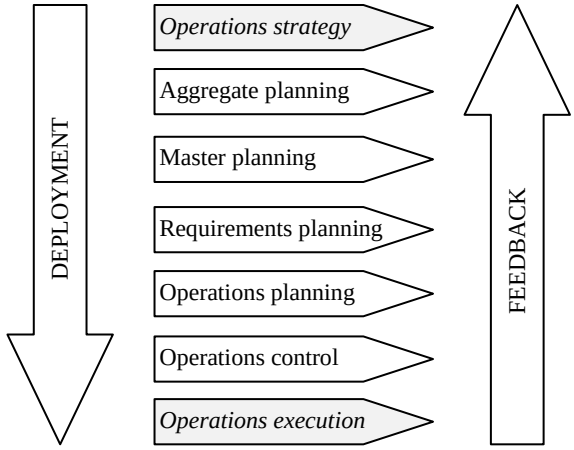


Fig. 18. Layers of the manage processes in manufacturing and logistics operations

The pattern presented in Fig.18 presumes that operations execution is controlled by operators or, in case of automated operations, by control units of devices, PLCs (Programmable Logic Controllers) or RTUs (Remote Terminal Units). Table 3 presents the key characteristics of all layers of planning and control processes in reference to discrete processes and pull flows, which is the most common case. To note, in case of pull-flows, some kind of self-control driven replenishment replaces the three lowest layers, like it happens in the Kanban control or within the 3C systems (Fernández-Rañada at al., 2000). It must be also noted that in automated cells and lines, the opera-

tions control is integrated with the control of operations execution, which is done by the means of MES (Manufacturing Execution Systems) (see also chapter 13).

The pattern presented in Fig.18 and also in Table 3 applies also to a significant extent to the process industries. The two major differences are as follows:

i. time-phased rates correspond to the schedules for batches;
ii. rate-based operations control and operations execution are performed under supervision of one control system of a facility/installation, which normally incorporates some MES functionalities.

Table 3. Characteristics of layers in manufacturing and logistics operations management (example of discrete processes and push flows)

Layer	Key function	Time horizon	Key outputs
Aggregate planning	Medium-term alignment of demand and resources	Few to several months	Aggregate master schedule
Master planning	Alignment of external demand with internal demand (requirements)	Several weeks to few months	Master schedule
Requirements planning	Alignment of internal demand with resources	Few to several weeks	Planned requirements (schedules, orders)
Operations planning	Scheduling of operations	Few to several days	Operations schedules or priorities
Operations control	Controlling operations (Shop floor control)	Ad hoc to few hours or one shift	Commands to execute operations

The layers of planning and control, as suggested in Fig.18, are integrated by two major activities, i.e. 'deployment' and 'feedback'. 'Deployment' refers to the deployment of business and operational objectives throughout the hierarchical structure of operations and logistics management. The operations strategy process primarily the feeds operations management by KPIs. The deployment at lower layers refers mostly to more or less aggregated and time-phased plans and schedules, which reflect the projected fulfillment of demand. The targeted KPIs are typically used as the means for performance control. However, as it was suggested in chapter 2, in the novel knowledge-driven solutions, we can imagine using KPIs directly for planning and control of operations (e.g. along the Input-Output Control at the 'Operations planning' layer or at the 'Operations control' layer). The 'feedback' activities support closed-loop recursive mode of planning and control. The feedback is obtained through:

i. the operations reporting, i.e. by aggregated records compiled according to the structure of plans and schedules; they are used directly along consecutive iterations of the planning and control activities, and respectively at their different layers;
ii. the performance measurement system; its results can be used to module some rules of decision-making, e.g. to change the priority rules according to current load of capacities.

The ‘deployment’ and ‘feedback’ activities are expected to integrate the neighboring layers, i.e. to support their coherence and alignment. Herein the issue of intra- and inter-layer integrity exhibits itself. It can be referred to various items, that are subject of planning and control activities at different layers, and determines the requested fits between them. The fits respectively refer to:

i. representations of independent demand (typically forecasts and customer orders);
ii. representations of dependent demand: requirements (plans, schedules), orders, and similar independent demand driven forms of plans and controls;
iii. flows of resources, material flows (deliveries, manufacturing batches, transportation batches), work flows (manufacturing orders, purchase orders, shipment orders etc.), data and knowledge flows;
iv. structural couplings of demand, products, processes and resources;
v. control and decision making models used for planning and controlling operations;
vi. performance targets, e.g. the KPIs (Key Performance Indicators), and/or state/phase or contextual transitions.

Granularity.
A particular issue when representing some aspects of the domain in different ways is due to granularity (e.g. of plans or resources), which is mostly driven by layering. It refers to many of the above listed items and primarily manifests itself by:

- Time-related granularity, i.e. due to different time units applied at different layers; it is reflected in: representations of demand, structures of plans and schedules and representations of requirements;
- Granularity of demand, i.e. with regard to spatial spread of demand and time-phasing;
- Granularity of plans and schedules, i.e. with regard to more or aggregate representation of products and due to time-phasing;
- Granularity of resources, i.e. with regard to different representations of resources concerning their structuring and functional abilities;
- Granularity of processes, i.e. with regard to different representations of processes, including their structuring and functional requirements;
- Granularity of material flows, i.e. with regard to different representations of material flows, e.g. more or less aggregated batching, like in reference to splitting manufacturing batches, lot-sizing etc.

The variety of granularities provides a separate factor of complexity, as different items managed at different layers have to be coherent along the time and consistent across layers. Therefore some fits have to be observed along operations performance and management, which are presented in the next sub-section.

Distortion and traceability of demand and supplies.
With regard to planning and control of operations the distortion of demand can be categorized in a twofold way:

- The backward (i.e. vertical) distortion of demand amplifies along communication of demand backward supply chain;
- The top-bottom (i.e. horizontal) distortion of demand amplifies along deployment of demand downward the hierarchy of manage processes, i.e. layer by layers.

Both types of distortion are primarily driven by some dysfunctionalities of planning and control, mostly in reference to:

- Time-phasing and offsetting demand;
- Netting of demand;
- Batching;
- Explosion of demand.

Additionally, dysfunctional (e.g. late, non-precise) feedback activities can amplify distortion of demand, i.e. along next iteration of planning at different layers. Distortion of demand is also driven by weak methods or models of decision making used for planning and control purposes. Granularity related incoherence may also handicap.

Distortion of demand is amplified by variability of operational processes, i.e. due to delays in the closed loop of deployment-feedback. However, distortion of demand amplifies variability of operational processes backward supply chain. Therefore a vicious cycle is driven be the two above factors. Altogether the unwelcome phenomenon results in a reduced efficiency of operations.

Following the above considerations two postulates can be proposed concerning future research and developments:

i. To tame the distortion of demand by relevant and appropriate ontological representations of the respective items;
ii. To support and facilitate novel methods and models of operational planning and control by the capacities of knowledge-driven management.

With regard to the explosion of demand, i.e. downward the layers of planning and control, the homogenous partitioning of demand and requirements protects against distortion of demand. Unfortunately it cannot always be applied. Nevertheless, it is a definite mean to protect the traceability of demand. A twinning issue is traceability of supplies. In some sectors this is a critical requirement. Examples are the aviation and the pharmaceutical sectors, wherein recording of the all tracking data is normally enforced by law.

Coordination and synchronization.
The execution of operations and the related manage processes brings about another type of challenging requirements, i.e. those related to the interdependencies arising along coordination or synchronization of concurrent activities. They are mostly centered around requested flow patterns or time-based coordination and synchronization, which could be eventually represented in ontologies, models and applications. Coordination and synchronization are also subjects of major interest of the computer science and automated control of manufacturing processes.

In operations management coordination is primarily supported by the layering of manage and control processes, which are basically integrated by procedures and communication of the 'deployment'-'feedback' loop of iterative planning and control. The existing solutions are usually weak and dysfunctional, as capacities of human performed decision making are limited, like it also happens with the support from traditional information systems. Hence, informal coordination is commonly used as an substitute, and an aid in case of issues, i.e. when some particular circumstances occur that were not considered within formal procedures. Altogether the ineffective coordination reduces economic efficiency, and results in many side negative effects, like poor customer service or turbulent behavior of supply chains. Therefore open knowledge-driven solutions are expected to improve coordination of operations.

It is an open question if the existing coordination mechanisms in operations management, when directly translated into ontologies and using the currently available abstractions and formalisms, will adequately reflect the 'logical' and 'real' nature of operations and manage processes. This question is especially relevant with regard to the environment of distributed or networked planning and control. Other solutions, e.g. those novel presented in chapter 2, like auctioning-based coordination, can induce even more challenging requirements. Hypothetically some inspirations can be derived from the theory of automation or computer science, e.g. from parallel and real-time computing. Strzelczak and Berka (2001) proposed a model of that kind for centralized coordinated planning of requirements and operations in distributed manufacturing systems, based on the Bulk Synchronous Parallel (BSP) model of parallel computing. Alternative novel solutions can be possibly derived by bio- and eco-mimetic ideations, like e.g. a coordination supported by the hormone-based controls.

Whatever are the coordination and synchronization mechanisms, in case of open knowledge-driven solutions they have to conceptualized by some meta-ontological means. This sets a question about existing knowledge on abstractions that could be employed when developing an ontology for manufacturing and logistics domain. Two streams of developments can be recalled with this regard. The first refers to modelling of manufacturing and logistics systems. Apart of various theories, which rather focus on particular functionalities of the models (e.g. theory of Petri-nets, theory of queueing networks, et al.), some of the developed tools may provide valuable insights, ranging from the 'old good' IDEF, to the recent ones, like e.g. the Unified Modelling Language (UML). The second valuable contributions were provided due to efforts aiming at categorization of some concepts, like e.g. workflow patterns, control flow patterns and so on. These works were too much extent driven by the Workflow Patterns initiative and resulted in a number of publications. The most representative were provided by Russel et al. (2005a, 2005b, 2006).

Coordination and synchronization are also among the basic and traditional topics of computer science. Operating systems, programming languages, user interfaces, and computer networks have to address the issue. However, the foundations of computing presume that the principal task of computers is just computing and data processing. Therefore the common abstractions of software consider the time as irrelevant (e.g. none of the common programming languages include temporal properties within its semantics). Consequently the prevailing view of time in computing is a bit peculiar,

which surprisingly applies to the domain of real-time programming. E.g. Wirth (1977) reduced its scope to dealing with threads with bounds on execution time, arguing that "it is prudent to extend the conceptual framework of sequential programming as little as possible and, in particular, to avoid the notion of execution time". In this sequential conceptualization, which founds designs of nearly all contemporary computers, programming languages and operating systems, computation process is accomplished by a terminating sequence of state transformations. Similarly, the classical approaches to concurrency in programming languages are based on extensions of the sequential programming paradigm. Therefore they are also weakly suited to meet the discussed challenges. And similarly, the two types of core abstraction do not meet requirements of real-time and concurrent coordination in operations management. Nevertheless, since 1980s experimental languages are being developed with temporal semantics. Most successful of them are those domain-specific, including both the modeling and the coordination languages, such as e.g. Giotto (Henzinger et al., 2001).

Coordination models and languages were evolving rapidly in recent years, as the concept of coordination, also with regard to SOA, is now adopted by many fields, e.g. middleware domains, modelling, and 'large-scale-coordination' applications, like the open distributed systems. The coordination solutions fall into two major categories:

i. data-driven;
ii. control-driven (or task- or process-oriented).

The main difference between them is that the last one exhibits almost full separation of coordination from process (computational) concerns. The state of any process at any time is defined only in terms of coordinated patterns that processes involved adhere to. Contrary to the data-driven coordination, i.e. where coordinators focus on handling data and checking data values, coordination and processes (computations) are treated like black boxes. The data handled by processes is of no concern to their environment, which is communicated by means of interfaces, i.e. between input or output ports. In addition to using ports, processes send control messages or events to their environment with the purpose of letting other processes know in which state they are. Hence, the coordination is driven by observing state changes in processes and broadcasting events.

The coordination concept in computer science is closely related to another one, i.e. of configuration and architecture description. Dedicated description languages for such purposes follow the principles of coordination languages. They typically view systems as complexities of components and interconnections, aiming at separation of structural and behavioral description of components. Formation of complex components is supported by hierarchical compositions of other components. Finally, they understand changing the state of some system as an activity performed at the level of interconnecting components rather than within the internal purely computational functionality of some component.

Additional amount of insights can be obtained from component software technologies, which have provided significant improvements in the area of large software systems. However, these technologies, including the object-oriented design and the service-oriented architectures, are also rooted in the sequential imperative and are

built on abstractions that match software and data structures rather than real processes or physical systems. They do not fit well to the concurrent or real-time computing, like to the requirements of manage and control processes in operations. Promising alternatives are given by actor-oriented component models and model-based design techniques. They often use formalization of the π process calculus. As yet they are mostly a subject of experimental developments. Some further discussion about SOA based Web-services, with regard to their orchestration, is presented in chapter 15.

The final comment is due to the general purpose networking techniques, such as TCP/IP, over which it is very difficult to achieve timing predictability. However, network time synchronization technology has improved considerably in recent years, so abilities of timing coherence across distributed computations were much enhanced.

The above review of computer science technologies exposes that even the recent developments are deeply rooted in some past but still dominant imperatives, like threads-based synchronization or sequential processing. They also exhibit a strong bias towards relational data-structures and object-oriented paradigm, which puts them away from the specifics of coordination and synchronization of distributed manufacturing and logistics. Hence they cannot adequately meet requirements of future coordination mechanisms, including those of operations management. Neither direct nor indirect fit between abstractions of the two coordination domains is within easy reach.

The problem of coordination and synchronization in automated control of operations is even more critical than in computer science or operations management. It particularly applies to open networks of embedded devices, which couple physical processes with computing. Considerable challenges arise due to concurrency, predictable timing and time-based synchronization, and unpredictable behavior of networked software, which cannot be effectively tested under all conditions. Today mechanisms of hardware interaction, e.g. those built on the concept of interrupts, are not well represented in programming languages as they come from operating systems, but not software (Lee, 2006). The common hardware coordination mechanism uses the abstraction of threads, which leads to incomprehensible and unreliable codes. To note, synchronous languages developed for safety-critical systems use more manageable concurrency model, hence they offer more predictable, controllable and understandable concurrency. Lee (2011) commented on Cyber Physical Systems development and stated that physical CPS components provide requirements, which are qualitatively different from those common in general purpose computing and object-oriented components programming. He suggested that to realize the full potential of future solutions it is required to rebuild computing and networking abstractions, which will have to embrace physical dynamics and computation in a new unified way. The outcomes from Claytronic project run at the Carnegie Mellon University, which focused collaborative interactions of massive numbers of smart appliances, have led to similar conclusions. Hence the Meld and LDP programming languages for massively distributed and concurrent systems were developed aiming at a more abbreviated syntax.

The discussion of insights for coordination and synchronization in manufacturing and logistics, which are given by the computer science and automation, reveals that representation of time and concurrency is weakly adjusted to the physical and logical aspects of operations. Although nowadays the upper levels of operations planning still

exhibit rather moderate time-related rigidness of synchronization, in opposition to the operations execution level, notably coordination provides a major issue and a big challenge therein. However, the situation in future production networks could be dramatically changed, e.g. due to the requirements of large scale solutions leveraging tracking technologies, which will surely exhibit properties of the distributed real-time control systems. Therefore the advantages of conceptual consistency of ICT and AT, and the framework for coordination and synchronization of operations, i.e. in terms of the grounding abstractions, cannot be overestimated. The abstractions and capacities offered by ontologies are surely among the sound options for answering the issue. With this regard the two major postulates can be proposed:

i. To develop relevant and appropriate ontological representations for the required functionalities and features of coordination and synchronization mechanisms;
ii. To support and facilitate the novel coordination mechanisms by knowledge models and knowledge-driven methods, i.e. through exploitation of adequate ontological representations.

Additionally, twofold R&D efforts concerning the ICT and operations management should address such topics as:

- incorporating time into models, programming languages and networks;
- fit of physical systems (hardware) and software in terms of grounding abstractions;
- support to predictable, schedulable, controllable and understandable concurrency;
- support to concurrent distributed real-time applications.

Spatial characteristics.
Spatial alignment of operations and resources, i.e. with regard to dynamic location of operations, provides another aspect of complexity along the execution of operations, and consequently to the manage processes. Semiotic interpretation of locations can encompass: physical locations (e.g. GPS address), structurally expressed locations, i.e. in terms of classes and objects (e.g. in reference to the layout or modularization of a given system), and denotations. Basically spatial alignment is limited by eligible fits, i.e. with regard to eligible flows (eligible physical paths for given flows), locations (e.g. in a warehouse) etc. The above mentioned complexities rooted in the spatial interdependencies, call for development of relevant and effective ontological representations of the spatial dimension of resources and processes.

Modularity and holonicity.
Modularity of capacities is a matter of particular importance in manufacturing and logistics. It is a direct factor of architectural complexities, this way conditioning operational complexities and some important performances, like especially with regard to:

i. volume of material flows;
ii. information flows;
iii. time rigidness and time criticality of controls.

The approach taken towards modularity may eventually facilitate the holonic features of modules.

The concept of modularity in manufacturing and logistics primarily applies to facilities. Module is a separate, lasting, architectural element of manufacturing system. It normally incorporates a set of manufacturing equipment, auxiliary devices or facilities, and eventually control devices together with HMIs. It basically provides an autonomous element of capacities or directly contributes to such a one. Therefore a given manufacturing or logistical system is built of modules, which are presumably featured with easy integrability. Consequently, it is expected that separate modules follow some predefined rules, what enables easy integration, e.g. by standardizing some technical or functional characteristics. The above also means that configuration or reconfiguration of a manufacturing systems principally involves modules, as the key (and normally only) elements subjected to any of the two activities. Modules are not only abstract constructs, but also separate physical entities, having own localization. They altogether compose a layout of a given system.

Modules are endurants. Semiotically module can be recognized as a quadruple of: real resources, a location, an ontological entity, and a denotation (e.g. URI, i.e. the Uniform Resource Identifier). Modules are presumed to have some kind of self-controllability, normally by the means of a single Remote Terminal Unit (RTU). They can be periodically fully self-sufficient, i.e. in terms of control and execution of processes. The concept of module in general corresponds to a subset of elements or components all controlled by one RTU. At best a given module may exhibit the features of holon.

The physical and control separations of modules are normally aligned. However, if this is not the case, the control consistency has prevalence over the physical consistency. This is in reference to the issue of disjoint nature of modules. E.g. in case of manufacturing line with buffers located between consecutive workstations, it is not obvious what should be the principle for assignment of buffers to the workstations, assuming all of them have some sensors and actuators. Finally, it might happen that while some modules will be clearly disjunctive in terms of rigid real-time controls, they do not need to be disjunctive in reference to operations control services. Hence, a strictly modularized control of operations execution may be merged with another one structure, i.e. heterarchical, which will refer to some other functionalities.

The links between modules and their elements are represented by:

- physical connections, i.e. in reference to the paths of material flows;
- informational connections, which may be eventually materialized by some physical informational buses;
- control connections, i.e. in reference to the cross-module control flows between the elements of control hardware (sensors, actuators, control units, PLCs, RTUs etc.);
- time connections; they have often to be taken into account, as control of operations is time-rigid and sometimes time-critical.

All of them may, and normally should be represented in an ontology, which would possibly support distinctions of modules. It is evident that such an ontology should be equipped with spatiotemporal and the mereotopological elements, respectively to

the nature of the above listed couplings. Alternatively, a graph representation of spatiotemporal mereotopological models could be also considered. It would help to better understand the complexities that condition separation of modules, and how the ontology behind this model should be eventually designed. Apart of that, it could be useful to analyze complexities, including assessment of different modularizations.

It is possible to extend the above approach to address the activities of operations control, i.e. the services. Structuring of services can be assessed in a similar way, i.e. like the variants of modularization. Furthermore, the two models can be integrated when needed, as a subject of assessment. The point herein is to make modules or services as self-contained, as possible. With this regard the four key performance measures can be applied:

i. metrics of material flows;
ii. metrics of information flows;
iii. metrics of computational loads (of RTUs and other hardware or services);
iv. metrics of time buffers and critical paths.

Although the above outlined idea of assessment directly addresses modularization of capacities and services, the proposed measures can be also used along adaptive control of operations, i.e. modulated by KPIs.

Other structural interdependencies.
Some other complexities can be identified with regard to the topic, which can be classified as structural. A common example refers to the planning and control models. Alignment of operational and performance targets provides a particular difficulty therein. A side issue is due to occasional events (breakdowns, blockings etc.) or eventual consideration of particular states (temporary overload or bottleneck, etc.) or other contextual factors, to enable adaptive management of operations. Therefore, relevant and effective ontological representations are required with these regards.

Functional characteristics.
It normally and often happens that along the operational processes, that a consonance of particular qualitative characteristics of some items has to be assured. Examples of such characteristics are: skills of operator, ability of a given machine tool to perform particular types of processes and within some range of parameters, ability of a stockroom cell to stock items with a given weight, ability of a software agent to run an auction of orders, product characteristics (e.g. with regard to their conformance with an available process or offered capacities), process applicability profile, etc. Principally functional profiles refer to products, capacities and processes. However their consonance is normally mapped through demands and requirements, i.e. along the manage and control processes (Fig.19). Matching of functional profiles is sometimes easy, e.g. when it is protected by a particular setting of machine tool, i.e. by its set-up. In other cases the functional conformance may require spatiotemporal reconfiguration of some resources, which obviously has to be preceded by some planning and control activities, and followingly by some actions at the physical level.

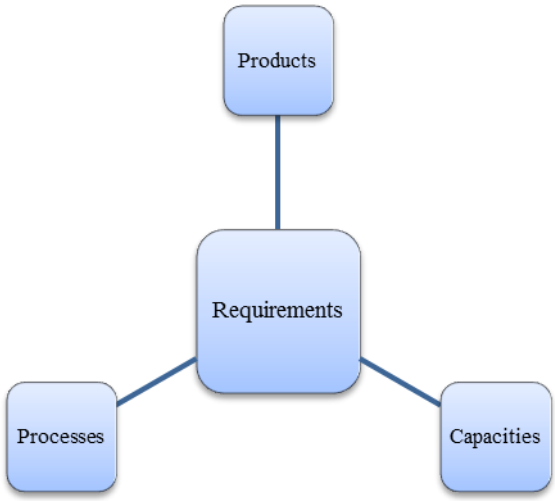


Fig. 19. The basic streams of mappings between the functional profiles

Whatever is the functional profile or its subject, it is normally expressed in three ways:

i. As a content, i.e. in a free descriptive way;
ii. As a collection of requested values of qualitative and numerical properties;
iii. As a structured mix of the two above.

The check of the fits between functional profiles can be performed in three ways, namely:

i. Check of identity;
ii. Check of similarity of content, i.e. by checking the semantic similarity of functional profiles (Resnik, 1999);
iii. Structural check of semantic similarity, i.e. by a qualitative checking of functional profiles which are represented in a structured way (Mormann, 2012).

The above discussion exposed the importance of an ontological representation of various items in terms of their functional characteristics. Such functional profiles are required to effectively align different items within the time and space dimensions, along the planning and control, and later on along operations. This also exhibits a requested tie between spatiotemporal and functional aspects within a relevant domain ontology.

Fits and constraints.
The preceding subsections revealed a plethora of complexities that normally arise along manufacturing and logistics operations, and along their planning and control. An adequate ontology, possibly featured with a simplicity, could significantly ease

knowledge driven operations management. It could also support ideation and implementation of novel systemic solutions.

One of the major sources of complexities are the particular aspects of various categories which result in some limitations which have to be observed along operations and their planning and control. The bird eye view of the identified types of constraints is presented in Fig.20.

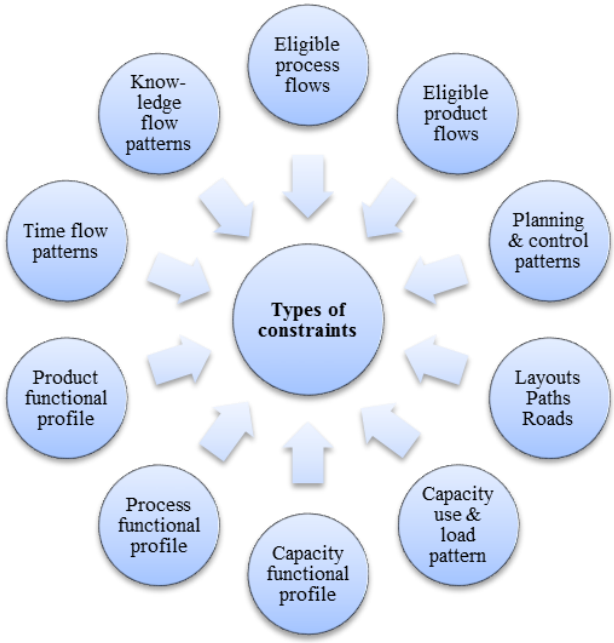


Fig. 20. Typology of delimiting constraints exhibited along operations planning and control

The observation of constraints, which normally happens in a run-time mode, leads to particular key fits, which have altogether to be met along operations management and execution. It can be derived from the earlier argumentations that in most cases the nature of complexities can be characterized in terms of the four dimensions, i.e.:

i. time dimension;
ii. space (location) dimension;
iii. structural dimension;
iv. functional (qualitative) dimension.

The analysis of all complexities results in distinction of conformance relations, which should be observed along operations management, within different layers of planning and control (following the commitment expressed in Fig.18, the execution layer is not considered herein). These relations altogether provide a concise but comprehensive perspective on characterization of various complexities. They can be clustered into separate items. Each of them represents logically and formally consistent composite

relation, which addresses one or more dimensions. Later on they will be referred as fits. The following types of fits can be distinguished:

- Spatiotemporal,
- Mereological,
- Mereotopological,
- Functional (qualitative).

Following the above categorization the first group may be separated, which refers to the aggregate planning. Two fits can be identified herein, which altogether focus on the rough alignment of aggregate forecasted demand with capacities. These two fits are depicted in Fig.21.

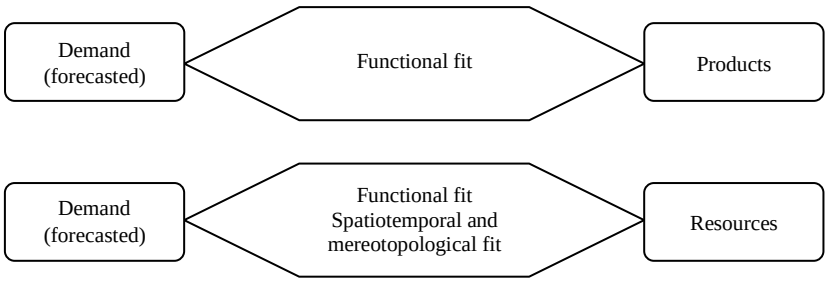


Fig. 21. Key fits along the aggregate planning

The first fit focuses on the functional consistency, i.e. of qualitative products characteristics with the expected demand. The demand is expressed in terms of aggregates of products. Products exhibit aggregated qualitative profiles, each of them expressed in functional terms.

The aggregated forecasted quantitative demand is also confronted with the load profiles of aggregates of resources, i.e. by rough-cut capacity balancing. This exhibits the second fit to be observed.

The next group of fits refers to the master and material requirements planning and to operations control. The focus of the first subgroup is on alignment of the gross demand with the dependent, i.e. internal demand, considering availability or resources. The distortion of demand is the primary concern on the way. The four fits of the first subgroup are exhibited in Fig.22.

The first fit refers to transformation of external demand to the requirements at different layers of requirements and operations planning and control. Requirements are expressed by spatiotemporal entities: plans, schedules, and priorities, like the master schedule, the material requirements plan, etc. Overlapping of time-phased demands of different granularity provides the major difficulty herein.

In parallel the second fit has to be respected, i.e. of consistency between planned requirements and resources. This fit is mostly rooted in qualitative consistency of requested and available capacities, and in time related coherence, i.e. with regard to the projected loads.

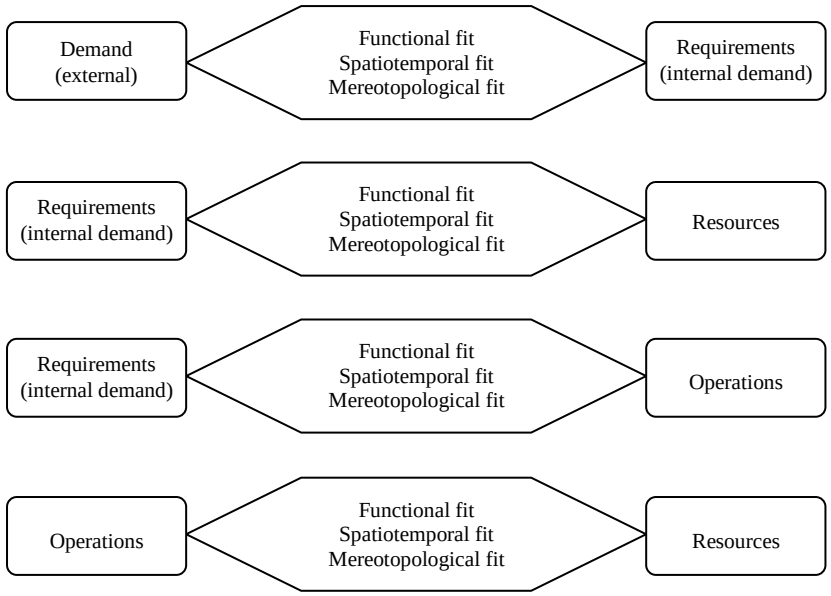


Fig. 22. Key fits along the explosion of dependent demand (requirements planning)

The third fit refers to the shop floor level of planning and controlling operations. The performed operations have to timely meet demand, as expressed at the upper level schedules and plans, while observing spatiotemporal, mereotopological (i.e. in reference separations and overlaps of phased and granulated demands) and functional alignment of operations and requirements.

The fourth fit applies to another aspect of the shop floor level of planning and controlling operations. Dispatching of resources and operations has to be coherent in terms of spatiotemporal and mereotopological fits, as well as concerning consistency of functional capacities offered by resources and those requested by requirements.

The second subgroup of fits was separated due to a very particular issue. Planning and control of operations has to consider a specific complexities, which are rooted in the structures of products and processes.

The structure of product can be in the best case reflected by subsumption hierarchy. Normally the mereological relations of parthood are better suited to represent such complexities. Apart of the elementhood or parthood relations the representation of a product has to exhibit functional profiles of its components.

Finally, the spatiotemporal interdependencies have to be taken into account. They are due to the necessary precedences exhibited by processes. The dyadic product-process representation, when well done, can provide significant advantages to the planning and control, especially with regard to the flexibility of operations. The mirror side of this issue is about a relevant representation of resources, to facilitate fitting of product-process dyads with resources. Such a fit in the best case has to consider functional and mereotopological aspects, while normally spatiotemporality cannot be ignored. All the fits concerning product-process dyads are exhibited in Fig.23.

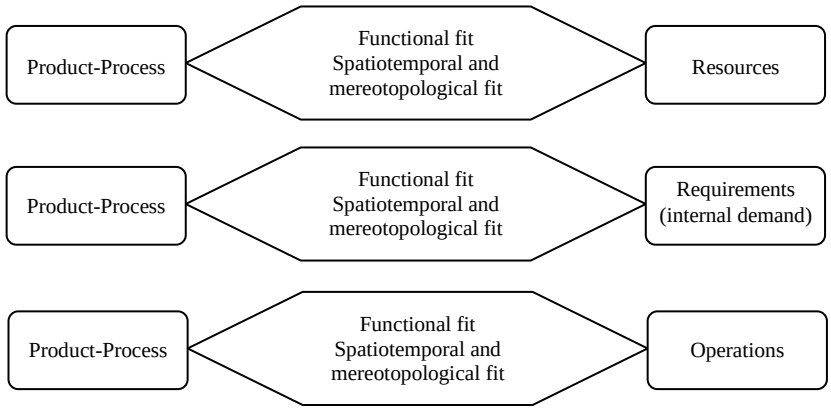


Fig. 23. Key fits due to complexities of the 'product-process' dyad

The first fit manifests itself along the downward acceptance of demand at some layers of planning and control activities. The granularity of functional characteristics of the resources has to be considered on the way. The second fit refers to those coherences of product-process structures, that have to be observed when exploding and time-phasing the demand, and then balancing it with capacities. Simultaneously functional coherence has to be observed. Similarly the last fit has to be considered when dispatching an operation takes place, i.e. often in the real-time mode. Like the previous fit, this one refers to coordination (or synchronization) activities.

The third range of conformance relations to be observed refers to the possibility of adjusting plans, schedules and disposals along adaptive planning and control, with regard to some discrepancies between operational targets and performance targets, like trade-offs. The welcome functionalities would be supported by some knowledge-driven capacities and conceptualizations, which could help to tame the conflicts of interdependencies and current targets. All these fits are depicted in Fig.24.

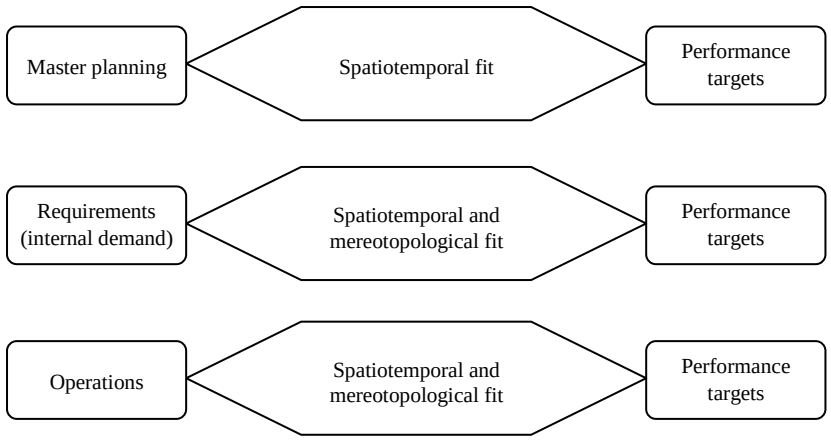


Fig. 24. Key fits due to KPIs

The set of fits presented above provides a transparent perspective on key complexities, which have to be met by manage and control processes in manufacturing and logistics. The proposed structure of fits was not aimed as a straight reflection of complexities of the existing operations management solutions, like those supported by the ERP or MES systems. It was rather expected to facilitate understanding of needs and requirements arising along development and design of ontologies, with a special regard to the expressiveness of ontologies. Support to development of novel systemic solutions was also considered as a welcome, but a side outcome. Nevertheless, the formulation of fits is directly helpful to development of ontologies.

5 Service Oriented Architectures, Web services, and Multi-agent Systems

It was argued in chapter 2 that significant advantages can be obtained by merging power of ontologies and SOA-based architectures for the open knowledge-driven manufacturing and logistics. This section discusses the possible alignments between ontologies and applications that could be understood in terms of the logical, architectural and operational coherence of applications (e.g. services, agents), and corresponding ontological components of the domain (e.g. resources, processes, products). With this regard a particular abstraction is proposed, which is centered around the transformational perspective on operations, and respectively manage and control processes. The transformation paradigm views all activities as a collection of concurrent and distributed transformations of resources and tasks, which are all coupled.

It must be underlined that application of ontologies for expressing semantics of production and logistical information does not restrict exclusively on Semantic Web or Web services (Vrba et al., 2011). The paper of Vrba et al. was among the first which demonstrated by a proof-of-concept how semantics and ontologies can be employed in industrial systems using the distributed agent-based solutions. The proposed approach was following those earlier research contributions, which were combining the agent technology and the ontologies with the Semantic Web services, into a coherent and integrated technology to support automated collaboration (Stollberg and Strang, 2005). Ermolayev et al. (2004) presented a framework for agent-enabled dynamic composition of Web services. The new provision of this proposal is in that the Web service is used as an agent capability supported with a relevant ontological description. The authors of this paper demonstrated how diverse Web services may be composed and mediated by dynamic coalitions of software agents who collaboratively perform tasks for service clients.

Nevertheless, many publications do not consider the conceptual merge of agents, ontologies and Web services, while rather relying on the two latter. E.g. Garetti et al. (2013) proposed a control architecture of automated manufacturing systems, which is based on the integration of the ontology, the embedded devices and Web services technologies. This approach allows substitution of the traditional control model, based on the hierarchical hardware architecture, with a new one based on a free series of pure software control levels. Therefore, a single-level population of controllers that

supervise various equipment at the shop floor level, is being managed via a communication network by the Web services.

SOA supports service orientation, which is a way of integrating ICT resources by interlinked services and the outcomes that they bring. Service orientation enables applications to invoke each other as a service, i.e. a repeatable task which meets some specified requirements, is discoverable, is self-describing and can be managed through some kind of governance (Fig.25). Whereas a software component is a unit of code that can be executed to provide functionality, a single service is a component that is actually running, often in its own process, which is hosted independently from the applications that are invoking it. Actually, applications themselves can be broken into parts that each run in their own process and invoke each other through services. Such an ideation makes possible composite applications, which are a set of linked services that support a particular process, e.g. a business process built on the SOA.

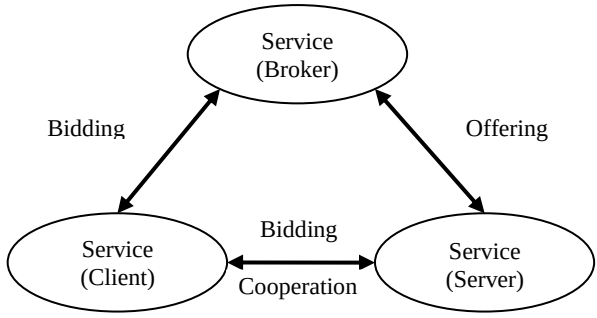


Fig. 25. Service Oriented Architecture – interlinked services

SOA is basically an architectural paradigm that considers mechanisms to publish, discover, coordinate and execute services (Fig.26).

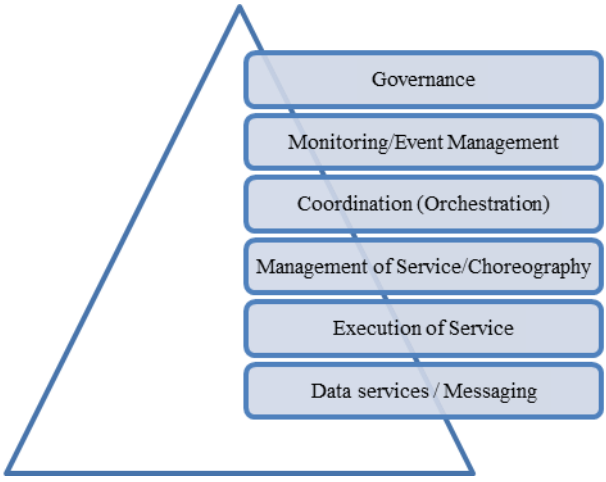


Fig. 26. SOA meta-architecture

According to Erl (2007) SOA follows the following principles:

i. standardized service contracts: services adhere to collectively defined communication agreements, i.e. service-description documents;
ii. loose coupling: minimized interdependencies;
iii. abstraction: beyond descriptions in the service contracts, services hide logic from the outside world;
iv. reusability: logic is divided into services with the intention of reuse;
v. autonomy: services have control over the logic they encapsulate;
vi. statelessness: services minimize resource consumption by deferring the management of state information when necessary;
vii. discoverability: services are supplemented with communicative metadata by which they can be effectively discovered and interpreted;
viii. composability: services are effective composition participants, regardless of their size and complexity.

SOA offers the potential to provide the necessary system visibility and device interoperability in complex systems, particularly those mixing the ICT technologies with some kind of automation, and subjected to frequent changes. SOA helps to put granularity of intelligence to the lowest level, i.e. of distributed intelligent agents, like smart appliances or embedded systems. Intelligent behavior at upper levels can be achieved by composite configurations, which provide incremental fractions of intelligence. This way self-diagnosability, adaptability and better performance can be facilitated. Furthermore, involvement of distributed ontologies enables re-configuration of loosely coupled software elements, instead of re-programming of large centralized hierarchical systems. Increased robustness and dependability of resources and processes could be indirectly but effectively facilitated this way.

The SOA is particularly applicable for such environments, which require frequent transformability and re-configurability. The manufacturing and logistics operations are this kind of environment. However, as an architectural paradigm, SOA does not provide a direct and sound response to the requirements of highly coupled, concurrent, and time dependent complexities of resources and processes.

The concept of Web services follows the SOA paradigm. The Web service is commonly understood as a software component, which provides a particular functionality, being not directly dependent on any platform or implementation. In the context of Semantic Web the Web service is embedded in a website. The Semantic Web is expected to enable users to discover, select, employ, compose, and monitor Web-based services automatically. The Web service is normally:

- Defined in a dedicated language (e.g. the WSDL (Web Service Description Language of W3C); to make use of a Web service, a software agent needs a computer-interpretable description of the service.
- Published and discoverable in a dedicated registry, using a standard mechanism; e.g. the UDDI (Universal Description, Discovery and Integration) XML-based registry for businesses can be used to discover services in registries or repositories, and define how they interact.

- Remotely invoked by a defined interface; to make use of a Web service, a software agent needs the means by which it is accessed.
- A part of other services, or a composition.

The openness of Semantic Web is not built on standard implementations or technologies, but on standard protocols. Therefore heterogeneous technologies can efficiently interoperate through shared protocols. This prevents from imposing standards by individual vendors. Open Source software development also plays an important role in preserving the interoperability of vendor implementations of standards.

The following standards play key roles for the Web services:

- UDDI;
- WSDL;
- Web Services Inspection Language (WSIL);
- Simple Object Access Protocol (SOAP);
- Web Services Interoperability (WS-I).

The interactions between the standards along the operation of Web services is illustrated in Fig.27. To note, alternative protocols can be used as replacements.

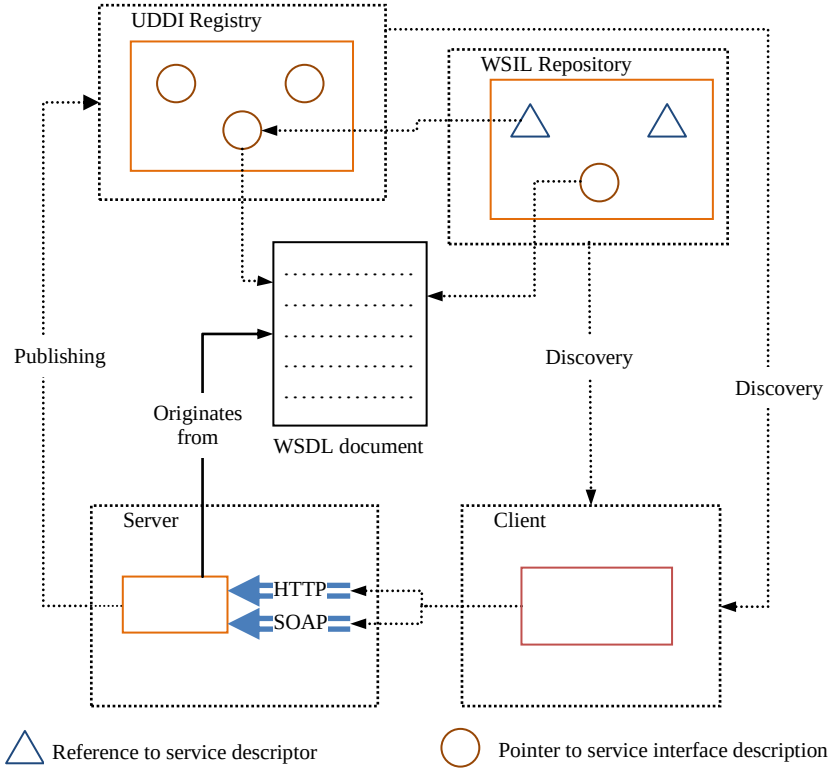


Fig. 27. An operational view of interacting Web service protocols [adapted from (Eclipse, 2009)]

WSDL is an XML-based open specification to describe interfaces and instances of Web services. The WSDL documents can be made available for their Web services through UDDI, WSIL, or by broadcasting the URLs to their WSDL documents via emails or Web sites.

SOAP is an XML-based protocol for accessing Web services by messaging over HTTP and other Internet protocols. It is used to exchange information in decentralized, distributed environments. SOAP enables binding and usage of discovered Web services by defining message paths for routing messages. SOAP may be used to query UDDI for Web services.

WSIL is an XML-based open specification. It defines a distributed service discovery method that supplies references to service descriptions at the points-of-offering of service providers. The discovery is explained by specifying how to inspect a Web site for available Web services. A WSIL document defines the location of a Web service description on a Web site. Since WSIL focuses on distributed service discovery, the WSIL specification complements UDDI by facilitating the discovery of services that are available on Web sites that may not be yet listed in an UDDI registry.

A server, i.e. the service provider, who hosts a Web service and makes it accessible from outside using such protocols as SOAP/HTTP or others. The Web service is described by a WSDL document that is stored on the server or in a special repository. The WSDL document may be referenced by the UDDI registry and WSIL documents. The WSIL documents contain pointers to the Web service interface description stored in the WSDL files.

The WS-I Simple SOAP Binding Profile and WS-I Attachments Profile are outlines of requirements to which WSDL and Web service protocol (SOAP/HTTP) traffic must comply in order to claim WS-I conformance. This is done by the Web services WS-I validation tools, which support WS-I SOAP Binding Profile and the WS-I SOAP Attachment Profile.

The typical scenario is decomposition of a complex task into a serial/parallel calls of various services, where data resulting from one service is used as an input to another service, or where data from many services have to be aggregated to create a response to the original request.

The aim of Web services is to provide a functionality, which can produce required data and effects in a widely accessible and machine discoverable form. The philosophy of Web services goes beyond the technical solution. The concept of Web services actually introduces a new paradigm affecting distributed computation and software system development. In the context of this new paradigm, the service composition stands as a mechanism combining multiple services to build up more complex functionalities. This enhances the potential of single services, like the overall one.

The combination of services may be required for two reasons:

- the user goal may consist of several sub-goals, each fulfilled by different service;
- execution of certain services could be required, to produce data or achieve effects which are necessary to execute other services; even if there is no single service providing the desired functionality, it may be possible to design a composite service from the single services, and thus provide the desired functionality.

The services are combined together using control constructs such as the sequential execution. The mechanism behind automatic web service composition must design a schema describing the control and data flows, i.e. it depicts the synchronization of the particular services, and prescribes which service produces output data used as input by other services, or achieves effect required before other services can be executed. Service composition usually does not design a time schedule. It rather focuses on the required inputs and the produced outputs and examines whether the outputs produced by a service can be consumed as inputs by another service.

Two other mechanism to operate multiple services in terms of their coordination are orchestration and choreography. (Web-)service orchestration presumes centralized coordination of services, while the choreography specifies peer-to-peer interactions between the services (see also chapter 15).

Several languages for semantic annotation of Web services were developed. They differ in complexity and expressivity. In each case the semantic annotation binds the elements of Web services to the domain terms. It typically focuses on the inputs and outputs, the conditions under which the Web service can be invoked (pre-conditions), and the conditions which hold after its execution (post-conditions). The OWL-S, promoted by the W3C, is an ontology of services that makes possible for users and software agents to discover, invoke, compose, and monitor services, making them interoperable and if needed, to do so with a high degree of automation. The OWL-S is supplemented by the Semantic Web Services Ontology (SWSO) (W3C, 2005).

The OWL-S ontology has three main parts:

i. service profile: for advertising and discovering services;
ii. service model: describes operation of services;
iii. service grounding: explains how to interoperate with a service via messaging.

The service profile explains ‘what the service does’, in a way suitable for any service-seeking or matchmaking agent to determine whether the service meets requested needs. It includes a description of what is accomplished by the service, limitations on service applicability and quality of service, and requirements that the service requester must satisfy to use the service successfully. The functional description of the service is expressed in terms of the data transformations produced by the service. It specifies the inputs and outputs, and preconditions required by the service.

The role of service model is to explain a client how to use the service, by detailing the semantic content of requests, the conditions under which particular outcomes will occur, and if necessary, the step by step processes leading to those outcomes. In other words, it describes how to ask for the service, and what happens when the service is performed. For complex services this description may be used by a service-seeking agent in at least four different ways, namely:

i. to perform a more in-depth analysis of whether the service meets its needs;
ii. to compose service descriptions from multiple services to perform a specific task;
iii. during the course of the service fulfillment, to coordinate activities of different participants;
iv. to monitor the execution of the service.

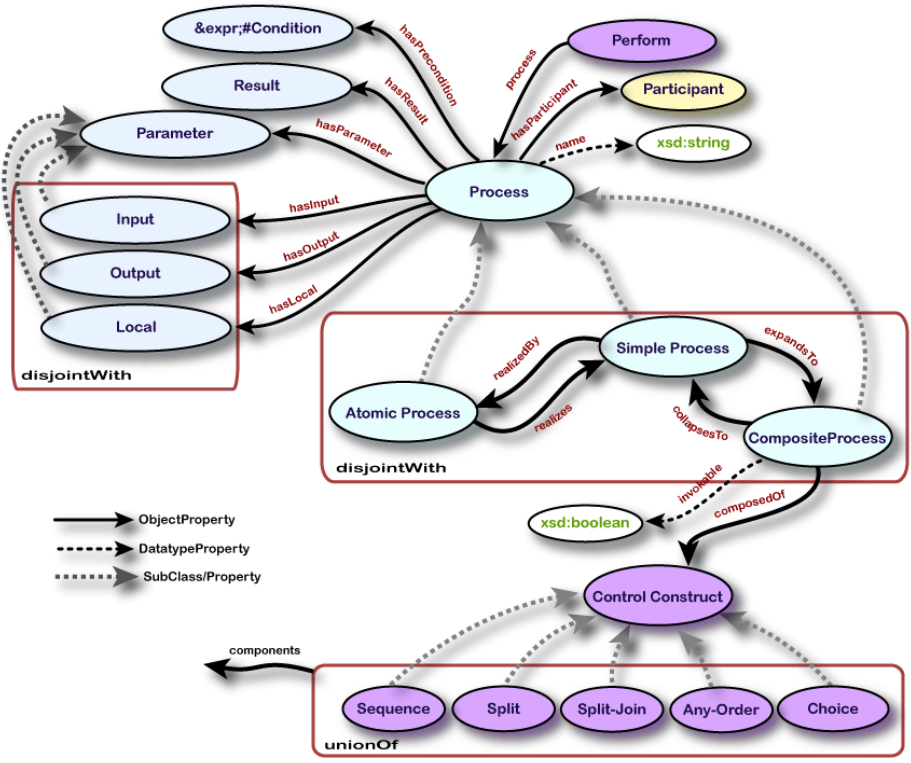


Fig. 28. The OWL-S process (sub-)ontology (W3C, 2004)

To give a detailed perspective on how to interact with a service, it is useful to view through the processes of interaction with the outside (Fig.28).

It is important to understand that a process is not a program to be executed. It is primarily a specification of the ways a client may interact with a service. Any process can have two possible purposes. Firstly, it can generate and return some new information based on information it is given, and the state of matters. The production of information is described by the inputs and outputs of the process. Secondly, it can produce a change in the state of matters. This transition is described by the preconditions and effects of the process.

An atomic process is a description of a service that expects one message and returns one message. Atomic processes are directly invoked, have no sub-processes, and are executed in a single step, i.e. they take an input message, run activity, and then return output message. Atomic processes correspond to the actions a service can perform by engaging it in a single interaction. An atomic process has always only two participants, i.e. the client and the server.

A process can often be viewed at different levels of granularity, either as a non-decomposable process or as a composite process. A composite process maintains some state, and each message the client sends advances it through the process. Com-

posite processes correspond to actions that require multi-step protocols and/or multiple server actions.

Simple processes provide an abstraction mechanism to provide multiple views of the same process. Simple processes are not invoked, but like the atomic processes, are conceived of as having single-step executions. Simple processes are used as elements of abstraction. A simple process may be used either to provide a view of a specialized way of using some atomic process, or a simplified representation of some composite process for purposes of planning and reasoning. In the former case, the simple process is realized by the atomic process, while in the latter case the simple process expands to the composite process.

Composite processes are decomposable into other non-composite or composite processes. The decomposition can follow specified control constructs. As many of such constructs in OWL-S took over the names of control structures from programming languages, it is easy to overlook a fundamental difference. A composite process does not represent a behavior of a service, but a behavior of the client along sending and receiving messages. If the composite process has an overall effect, then the client must perform the entire process in order to achieve that effect. One crucial feature of a composite process is its specification of how its inputs are accepted by particular sub-processes, and how its various outputs are produced by particular sub-processes.

Apart of service description ontologies and languages, dedicated service composition languages have been developed to support (Hahn et al., 2013). An example is the WS-CDL (Web Service Choreography Description Language) of W3C. However, most of the Web service standards and service composition languages do not deal with the dynamic composition of existing services. This challenging problem is common in the practice, e.g.: in automated materials handling; when a functionality that cannot be realized by the existing services is required, while the existing services could be combined together to fulfill the request. As yet this area of R&D focuses rather isolated issues. To draw the whole picture of the problem and make dynamic service composition a reality, a number of problems have to be addressed, especially with regard to abstractions and formalization which found the existing languages and tools (Bartalos and Bieliková, 2011).

The Web services implementation can be supported by many available resources. Therefore the major effort required along applications development was shifted from the coding to the capturing of logic of the business processes. Although the SOA approach poses many advantages, it actually focuses interactions among the information resources and the users of data-driven services. This leads to particular way of abstracting about processes and resources. With this regard, and especially considering the topic of this chapter the following question could be given: ‘Do the concepts of Web services, like the W3C standards, fit well to the conceptual needs and requirements of manufacturing and logistics?’. A reverse question could be also asked, if operations, and operations management and control, can be effectively conceptualized in terms of the Web services and related ontologies, respecting the assumptions of SOA. A simple and example test-of-the-concept can be performed by using the OWL-S process ontology to represent a simple case of manufacturing processes.

It often happens that a manufacturing order, i.e. a batch, has to be split into two parts. Normally it takes place before the original order was started, and the only possible way to meet the circumstances, is to close the existing order, and to issue two new orders, which would normally inherit the original process (typically a standard one). It appears, that conceptualization of such a situation in OWL-S is not possible. The process model consider split of process, while split of services is not possible in a direct way. Evidently the abstractions behind the process ontology are not suited to common realities of manufacturing and logistics. Of course it is possible to meet the circumstances by the OWL-S, however paying a good price for increased human effort and decreased efficiency. The above example was chosen due to its simplicity and transparency. Many more examples of that kind could be given in reference to OWL-S and other tools used to develop the Web services. Not to forget, the OWL-S is not directly suited to description of the spatiotemporal and mereotopological conceptualizations.

The above case exposes the importance of founding abstractions and grounding theories, as well as of matching between the reality of operations, the manage and control processes, and the ICT & AT processes. Operational resources and processes should be abstracted in a particular way to be represented through services, which are principally viewed by the SOA as information processing based activities, while the nature of real processes is not considered by the Web services.

Interestingly, the example exhibits another logical incoherence of common conceptualizations of services and operations, and those of SOA. The operations are primarily driven by some kind of requirements or demand. These are subsequently exploded to plans and schedules, in the meantime being offset or batched. Finally simply steps at the shop floor level have to be executed, according to some kind of demand or triggering, and with respect to some control rules. The logic of operations is consistent with the logic of (Web-)services. However, this conformance does not enfold formal foundations of existing WS SOA-based standards and tools. The exposed drawbacks can be eventually overcome by the conceptual abstractions of the transformational approach, which will be presented later on in this section.

Most of the futuristic ideations presented in chapter 2 cannot be implemented before massive software-integrated and -controlled industrial environments become realized. Typical standalone and compartmentalized solutions need to move toward distributed, decentralized and networked architectures with intensive communication and collaboration. In such complex environments in order to operate efficiently, a high level of mutual understanding is necessary along decentralized decision making and control. This translates into a reasonable understanding of domain-specific semantics, and the Multi-Agent Systems (MAS) are among the most promising and natural realization choices, especially in conjunction with the prevalence of service-oriented architectures and Cloud technologies (Colombo et al., 2015).

Agent technology is utilized to develop such systems, where autonomous agents execute their specific tasks in a collaborative manner. The interactions between agents may vary from simple data exchanges, through requests to perform a particular operation, to complex negotiations, e.g. based on different auction mechanisms or the contract-net protocol. Agents have distinctive inherent properties, namely:

i. Autonomy: an agent is capable to solely act on its specific task, i.e. without any direct intervention or support. It has some extent of control its internal state and acts on tasks using its own capacities.
ii. Reactivity: an agent is capable to respond to events and changing circumstances in its environment.
iii. Recurrence: an agent is active by continuously running processes. When the current processes have been completed, then it is reactive for another ones.
iv. Collaborativeness: interactions between agents are appreciable as partnerships for performing their tasks. In case of the multi-agency, agents work in a continuous cooperation. Some matchmaking mechanisms are commonly used for collaboration purposes.
v. Purposefulness: an agent has specified scope of goals in its applications, and it is by its nature goal-oriented and purposeful.
vi. Mobility: by mobility of an agent it can make the task easy over network. An agent can move and run in different networked locations.

As distribution of decision making is presumed in industrial deployment of the agent technology, the manage and control processes can be spread out among a community of autonomous and mutually cooperating agents. At the shop floor level an agent may represent and independently control particular physical equipment, like an RTU in the eScop approach.

The MAS paradigm merges the agent technology and semantic technologies. With this regard the integration scheme of ontologies and application models presented in section 3 (Fig.16) can be considered, e.g. by a mechanism of updates of agent behavior (i.e. its program code) as the ontology is being extended. The corresponding piece of code providing an agent with algorithmic description of its process associated with a new component of the ontology is sent to the agent so that it can subsequently execute the code to react appropriately. The MAS paradigm replaces the centralized control by a distributed one. Due to that the interactions of individuals may eventually exhibit an emergent global behavior, possibly some kind of externalized intelligence. This allows to reach a high degree of autonomy and cooperation, not like in a fixed client-server structure. The MAS concept supports decentralization of control, as well as modularity of applications, making them self-contained, intelligent, proactive, changeable, robust, fault-tolerant and reusable (Unland, 2015).

It is suggested in the literature that agent-oriented control systems and agent technology may help to cope with the high complexity of planning and control activities along manufacturing and logistics operations (Vogel-Heuser et al., 2015). Intelligent agents can be efficiently used for planning and control of operations, and the services provided by agents can be described in the OWL-S, like the Semantic Web services (Pěchouček, 2008). However, MAS principally has no controller and no means to take a global perspective, e.g. to control global homeostasis or performance (Jennings and Wooldridge, 1998).

Many similarities exist between MAS and SOA (Mendes et al., 2009). Therefore a high potential of synergy between these approaches may arise when both are merged. One of the basic concepts of multi-agent systems, i.e. the service matchmaking, pro-

vides a good argument for that. The basic mechanism for service matchmaking is based on publishing skills and services by agents. It can be applied in an ontology-driven mode, e.g. matchmaking of services can be supported by the OWL reasoning.

In reference to the above comment an example solution based on multi-agent technology can be given for the shop floor control activities. A community of interacting agents, who altogether represent service providers and requestors, can be nested so that demanded complex services are coordinated as compositions of basic simple services, and orchestrated. An important functionality herein is the distribution of tasks over multiple agents. This goes beyond a simple service matchmaking, as the whole process must be decomposed, executed, and the arising problems have to be resolved in a runtime mode. Using the ontological representation of manufacturing processes, which is an obvious option herein, an automatic discovery of the equipment capable to perform requested operations should be enabled.

The above example exposes the need for conceptual matchings and mappings, which has to be satisfied along the development and operation of ontology-aided solutions. The first perspective of them is rooted in the overall view of the domain, which enables distinction of the primary, secondary and tertiary processes. Fig.29 provides a comprehensive but simplified view of the conceptual couplings between these processual subdomains. All of them should be reflected in both, the meta-ontology and the core domain ontology. Otherwise the conceptual inconsistencies may deteriorate the development of ontology and the operation of ontology-aided solution.

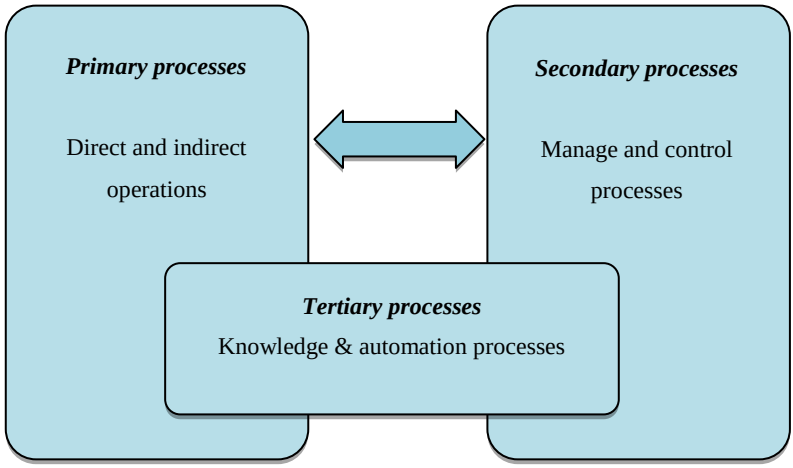


Fig. 29. The requested conceptual matching of the three processual subdomains

The taxonomy of processes is based upon the distinction of direct operations and indirect operations. The direct operations are just the all operations that add or deliver value to the customer. The indirect operations straightly enable, condition or support the execution of direct operations. Both types of operations compose primary pro-

cesses. Examples are: fabrication, assembly, customer processing, engineering, logistical processes, staffing, quality assurance, payments, and so on.

The manage and control processes, i.e. the secondary processes, are super-ordinate to the primary processes. The concern herein is how to organize, balance and coordinate the flows of primary processes. Common examples of such processes are: planning, scheduling, controlling, prioritizing, dispatching, synchronizing, monitoring, feedbacking, correcting, recording, reporting.

The tertiary processes are operated by the means of KCT (Knowledge and Communication Technology) and AT (Automation Technology). They are nested in some ICT or AT components, e.g. agents or services. Therefore they have to be supervised, driven and supported by some other processes, making them alive and harmonized. Such tertiary processes are actually centered on how to maintain, and primarily how to trigger, the automated manage and control processes and the execution of operations. Referring to the Web services, example tertiary processes are composition, orchestration and choreography (see chapter 15).

The tertiary processes provide a knowledge and automation infrastructure to the others, including singular operations, operational processes, and operations as a whole. They deliver knowledge and information provisions to support execution of singular operations, and to support or automate the manage processes, including the decision making. Although they are embedded in the primary and secondary processes, and overlap them (Fig.29), there are particular reasons to separate them, which actually has nothing to do with the technological aspect.

The tertiary processes often operate themselves, or autonomously operate the secondary and tertiary processes. Sometimes they are externalized. This trend is visibly enhancing, and the quick development of Web and Cloud services provides an excellent illustration for this thesis. It often happens that tertiary processes enable novel systemic solutions of creating and delivering value to the customers. Some existing developments, e.g. the crowdfunding portals, suggest that tertiary processes tend to merge with product and funding markets, and visibly this trend leads towards new emergent forms of economic institutions. Therefore the role of tertiary processes is crucial, contrary to the semantic message from their name. Finally, the technology aspect helps to clearly separate the area of tertiary processes.

Another view on the required conceptual matchings and mappings is obtained by the distinction of technological components of tertiary processes, which are:

i. the ontology itself, as a representation of the whole;
ii. the agents who provide services;
iii. services that transform the resources;
iv. the resources;
v. the tasks to be executed, which exhibit the mirroring of required services.

Fig.30 provides a comprehensive bird-eye view of the conceptual couplings between the technological components. Ontology represents all systemic aspects of the ongoing activities, and itself. The architectural and operational mirroring of tasks, resources, agents, and services is made possible by the means of local sub-ontologies. Similarly, focused dynamic matchings and mappings (i.e. spotlight views) between

the overall ontology, and its localized components, can be realized in a runtime mode by a variety of captures and provisions. This approach is logically consistent with the eScop approach.

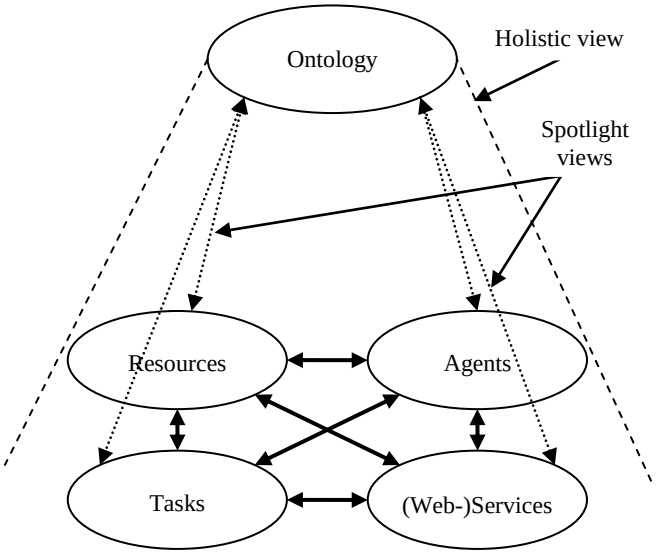


Fig. 30. Adjacent and transverse matchings and mappings in ontology-driven solutions

Whether and how the overall ontology can be fully split into localized components, is a separate issue. Ontology agents can eventually provide a substitute solution, i.e. by the means of modularized ontologies.

The above review of SOA and MAS, in reference to the ontology-aided solutions, confirms that the merge of three paradigms may provide significant advantages. However, as it was exposed in reference to the specific requirements of the domain, the existing developments and technologies have to be revised. The key reason for that is the need to improve the logical coherence of the grounding meta-ontological abstractions, and the peculiar aspects of the domain, especially the dynamical one. Such a coherence is crucial with regard to the following qualities:

i. convenience, i.e. with regard to intelligibility;
ii. expressiveness;
iii. reasoning capacity;
iv. performance of the solutions.

The above questions are addressed in the next sections.

6 Transformational approach to ontologies for manufacturing and logistics

Following the discussion from the preceding section, as well as the conclusions from earlier sections, the transformational paradigm can be introduced now. It is targeted as a unifying abstraction of operations, manage and control processes, and ICT & AT infrastructural processes, i.e. driven by the service- or agent-based applications.

Transformations should be viewed as an abstracting vehicle for the holistic conceptualization of the whole domain, which is primarily viewed as the couplings of resources and performed tasks, which are altogether tied with the ongoing transformations (Fig.31).

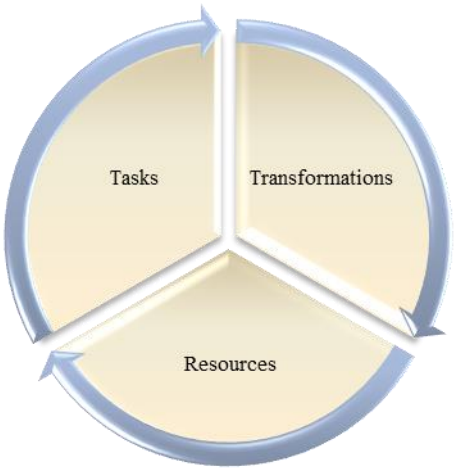


Fig. 31. The spinning wheel of 'Tasks-Transformations-Resources'

The transformational paradigm is built upon the fundamental conceptual construct: the 'tasks, transformations and resources' trinity. Altogether it describes the change of everything, along the flow of matters, which goes along the flow of time. The 'everything' composes a logically coherent operational environment, like an ecosystem. In particular case it may be a system or an economic institution, e.g. a virtual market.

Tasks provide the vital power to the transformations, which represent the flow of matters, i.e. and the life of ongoing operations. Tasks are expected to bring some outputs or accomplishments. The consecutive tasks are triggered by the outcomes of the preceding tasks. Tasks are fed with resources, which are the matter and the subject to transformations. The transformations consume, use or change resources. The consequence to them is the changed state of matters.

Transformations change resources and vivify tasks. The basic type of transformations in manufacturing and logistics, are the operations (processing transformations), i.e. when materials are processed into products, and the logistics transformations, i.e. when goods and some equipment are reallocated and delivered to their

destinations. To run transformations, e.g. to execute a single manufacturing operation, capacities may have to be temporarily configured, like workplaces, staff, tools etc. Therefore they are also a subject of spatiotemporal, structural and functional transformations. Hence, resources are inputs and outputs to transformations, both the material and the virtual ones.

The SOA actually focuses on transactions between service providers, while the essence of services, i.e. transformations, is practically disdained. Unfortunately products and production processes, and the demands behind them, even if decomposed and distributed, are still strongly tied. Therefore the relevance of the SOA abstracting about capacity is doubtful in case of the highly coupled environments. The manufacturing systems and supply chains are good examples of that kind, especially when products are engineered to order (ETO).

The abstraction of resources extends the common understanding of this term. Anything that supplies a given transformation, or outcomes from it, is actually a kind of resource, that was indispensable, or will be later utilized. Even a simple data, which is a carrier for some control command, may be a critical resource for some transformation. Consequently, although the resources are a matter of transformations, they can be logically recognized as the inputs and outputs (Fig.32).

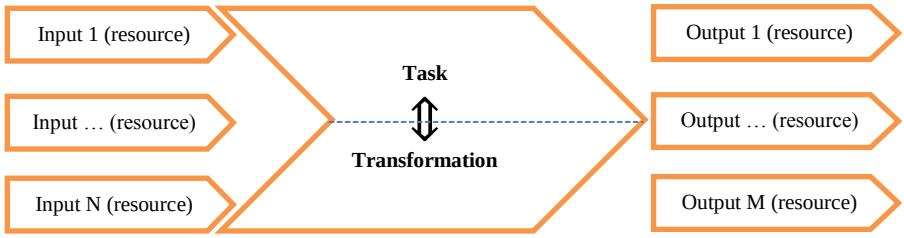


Fig. 32. The operational-constructional view of the 'tasks-transformations-resources' triad

Tasks, transformations, and changing resources trigger each other. Altogether they compose an externally triggered and self-triggering life. The market demand is basically the primary source of tasks in manufacturing and logistics, which are then transformed to the requirements (dependent demand), later on to the orders, then to the disposals of operations, finally machine control units may take over their jobs.

Apart of demand-rooted tasks, others are generated in line with the latter, or due to particular settings in the environment. E.g. when idle, the smart device may call for a task. In case of breakdown of a particular machine an agent managing the manufacturing cell may set an outsourcing task for manufacturing services. When accumulated operation time of some machine exceeds the assumed resource, or when its operating parameters drop down below some limit, a relevant agent may release a call for maintenance service. In case of anticipated shortage of capacities along the aggregated planning, the planning agent may call for extra shifts. These few examples illustrate the reach of mechanisms that could be eventually employed to drive those tasks that do not directly relate to the market demand. Most of them support some manage processes that prospectively harmonize the demand, the operations and the resources. Others may refer to security or safety issues. Altogether they provide the homeostatic

functionalities to the ecosystem of manufacturing and logistics operations. The above discussion focused on real tasks and transformations. The dual question is about the technology for driving the stream of tasks. This issue will be discussed later on.

By their nature transformations are continuous or discrete, like it happens with the discrete and process manufacturing and logistics (see chapter 1). The transformational paradigm can be fit to the both types of transformations. Furthermore, transformations can be by their nature qualitative or quantitative. The transformational approach recognizes the both types of transformations. In particular, the transformational paradigm can be unwrapped to address the following mechanisms used to trigger tasks:

- Timed triggering:
  - By schedules;
  - By timed triggers (e.g. timed events);
  - By periodic triggering (e.g. periodic replenishment);
- Non-timed triggering, i.e. relativistic triggering:
  - By non-timed and non-scheduled triggers:
    - Non-timed events;
    - States;
    - Phase transitions;
    - Non-periodic replenishment;
  - By phenomenological triggering, i.e. by coupling variables of parallel or sequential continuous processes with functional interdependencies; these may be represented in two ways:
    - By constructs of non-timed functions, functionals, and equations;
    - By constructs of semantic interdependencies, which in most general case can be expressed by the means of prototothetic functors.

The transformational paradigm constitutes three basic constructs to abstract in the domain of manufacturing and logistics. In terms of the theory of categories and functors, they can be refereed as categories or functors. This theory principally separates its two basic abstractions, i.e. the categories and functors, while herein the reality is considered in terms of spatial or spatiotemporal trajectories, which are all coupled by the functorial equivalences, as defined by Eilenberg and MacLane (1945). Furthermore, the paradigm applies to the nonlinear ecosystems, in the Yoshida's sense (Yoshida, 2008), by conditioning the ongoing changes in response to the changing condition of the ecosystem. Such a mechanism does not necessarily be consolidated in a single parameter. This feature is supported by the overall concept of triggering mechanism. In reference to the manufacturing and logistics ecosystems, it can be versioned in a way illustrated above, while an ontology supplemented with a prototethic, can operate as a vehicle for qualitative and quantitative handling the three abstractions. A full and in-depth discussion of these question, due to its meta-scientific and meta-mathematical character, goes beyond the scope of this book.

Tasks, while driving transformations, undergo transformations along their completion. At the early stage of their life-cycle they may be subject to design. Referring to the example presented in Fig.14 design of picking task means, that an order for some items to be picked from the stockroom has to be constructed. The prior input to the

design of task, is a pool of requested stock items to be disbursed from the stockroom. The output of the design will be later used by a software agent, to control switching-on and switching-off the relevant lights, this way instructing the operator. To design a task the ontology of stockroom layout, paths, and the inventory data, have to be used.

Tasks interact each with other. Three patterns of tasks triggering (creation) by another task can be distinguished (Fig.33). Completion of one task may deduce another task, i.e. through a particular output of the former one. Oppositely, a task may be induced by an input to the given task. E.g. the requested configuration of equipment, as an input to a given task, induces another task, which should result in some configuration the required resources. Both cases exhibit two ways of triggering tasks, which are input- or output-driven. Tasks may be simple entities, or composite structures of subtasks. This provides another way of triggering tasks, i.e. through explosion. The parent task may eventually invoke some child tasks. Inheritance of inputs and outputs along the creation of new tasks is also illustrated in Fig.33.

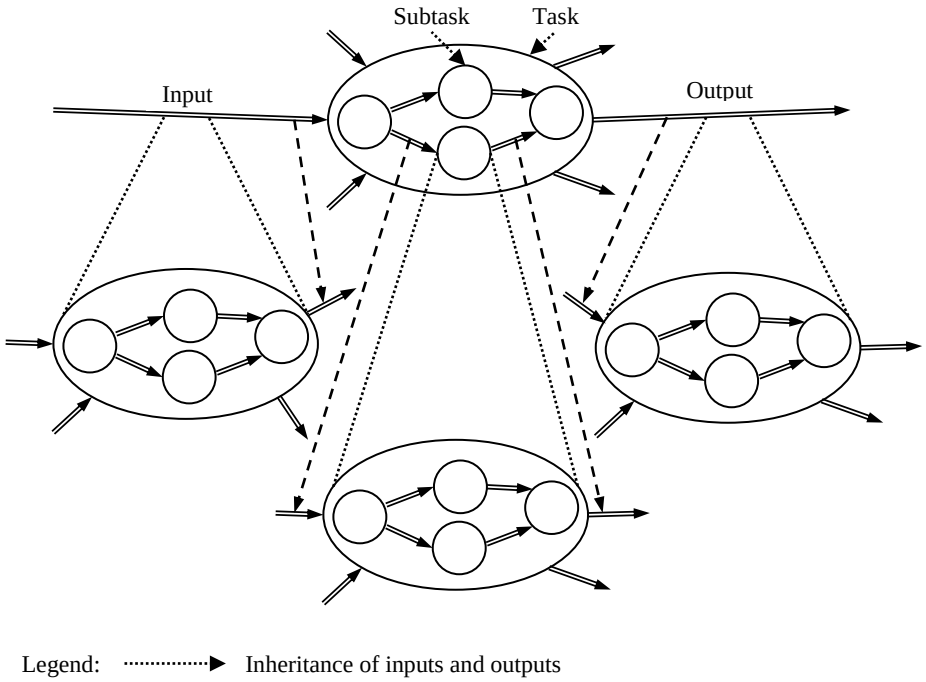


Fig. 33. Mereotopology of tasks, transformations, and resources:
invoking tasks as induced input tasks, deducted output tasks, and exploded composite tasks.

The picture actually visualizes a networked mereotopological structure of nested inputs and outputs, i.e. resources, and tasks. All of them have altogether to be represented in an ontology, which is also expected to provide those patterns, which are indispensable along the tasks creation and design.

Another opportunity of triggering tasks is by an aggregation of some subtasks from different parent tasks. The scheme of aggregation, including the inheritance of inputs and outputs, is presented in Fig.34. Alternatively this scheme can be applied to aggregation of tasks from inputs or outputs. E.g.: few shipments may be merged into a new one; two manufacturing batches can be aggregated into a one task of heat treatment, performed by a furnace. Altogether three aggregation patterns for tasks have been distinguished.

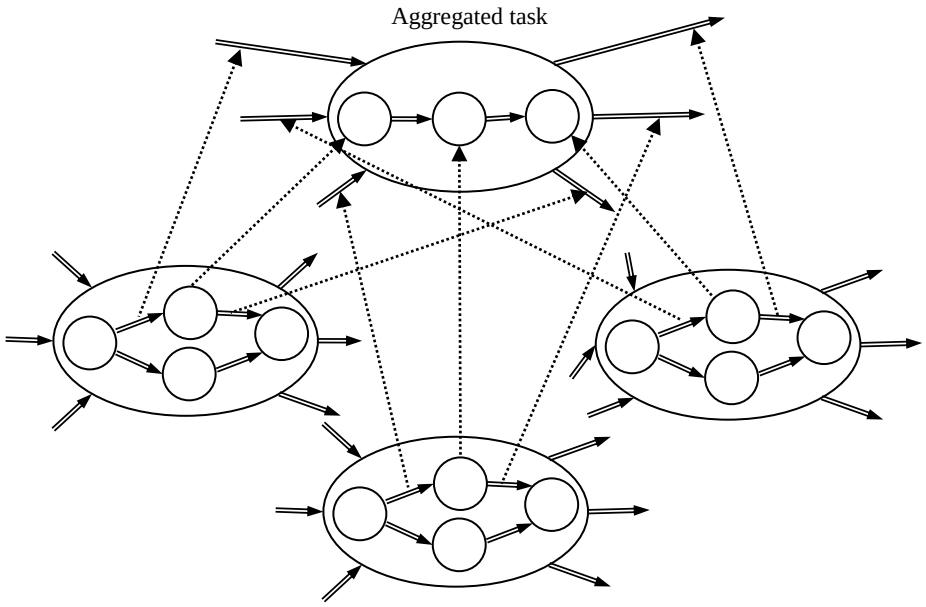


Fig. 34. Mereotopology of tasks, transformations, and resources: aggregation of tasks

Many publications addressing automated coordination of Web services, i.e. orchestration, presume hierarchical structuring of services during their composition. The above examples, as well as the two schemes expose, that in many circumstances tasks, which may be considered as analogues of services, will exhibit non-hierarchical structures. Despite that the controllability of all tasks, and the ability to coordinate them, can be in most cases fully protected. Of course, as was suggested in the section 5, the parallel issue of traceability of demand and supplies, may coerce to particular tracking measures within an ontology.

The simple example presented in Fig.35 provides an argument in favor of the above stated thesis. The batch of six items is twice split, and then merged again, when flowing between the three consecutive tasks, according to the available transportation capacities. The controllability of tasks, i.e. in terms of ability to coordinate them, is

easily assured herein. It is due to homogeneity of transformations of tasks, which is protected locally. Therefore handling of hierarchical structure of tasks does not provide an issue, as the problem is 'localized'.

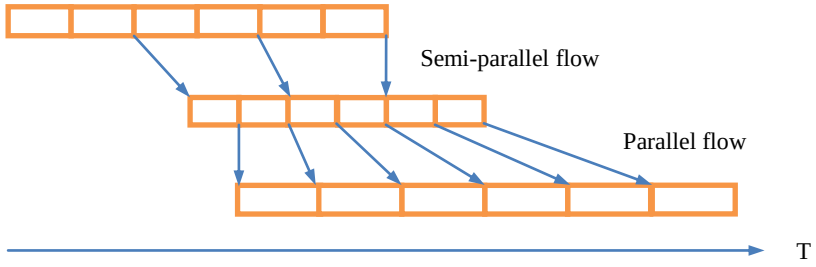


Fig. 35. Homogenous partitioning and merging of tasks along parallel and semi-parallel flows

The coupled transformations of tasks and resources can be conceptualized from the two perspectives, i.e. the behavioral one and the representational one, as it is depicted in Fig.36. The concern of behavioral perspective is about how things happen, i.e. how they are operated, managed and controlled, and finally executed. Different viewpoints can be distinguished, which separately and respectively address the primary, secondary, and tertiary processes.

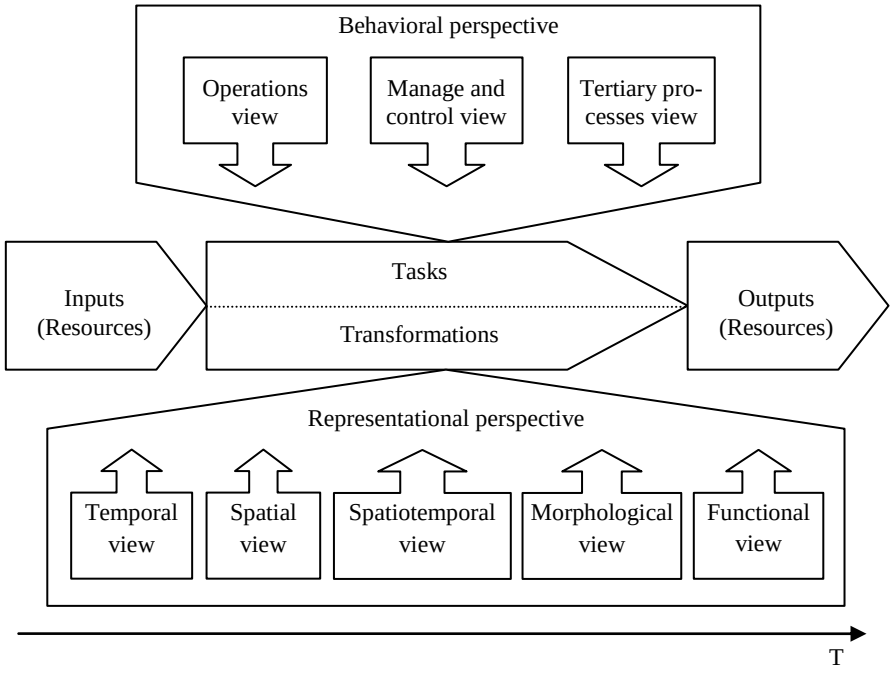


Fig. 36. Behavioral and operational perspectives for operations management

The representational perspective focuses on how the changing state of matters is conceptualized. Five integrated dimensions of change can be distinguished with this regard:

i. time;
ii. spatial;
iii. spatiotemporal;
iv. morphological;
v. functional.

The temporality can be simultaneously represented in different ways. Therefore miscellaneous allocations of different items along the time scales are enabled. The time flow can be layered, and more or less granulated, e.g. using different time units and calendars, both integrated by the partitioning relations. Apart of the time-based ordering (timing), the ordering by relativity of change, complements the temporal view of flowing matters. Other conceptualizations of temporality may also apply.

The flow of matters relates also to the spatial relocations, which primarily refer to the Euclidean space, or to the geographical space. Other concepts of location are often applied, particularly with regard to the semiotic interpretation of locations.

The spatiotemporal view encompasses those entities and aspects, which simultaneously manifest the spatial and the temporal nature. This is the case of all fundamental categories, i.e. tasks, transformations, and resources. An example of spatiotemporal space is the space of manufacturing orders, if considering their status, i.e. the state of progress. Similarly space of resources and their states can be distinguished. E.g. using the latter concept, phase transitions can be recognized for the advantage of adaptive planning and control. Spatiotemporality can facilitate timed coordination. For instance some timed synchronization mechanisms can directly take advantage of temporalities. The timed coordination can be supplemented or substituted by another one, which may possibly exploit the indirect time-related orderings, like e.g.: task-driven, state-driven, event-driven, precedence-based, co-occurrences, situation-driven, hormonal et al. The spatiotemporal change may also refer to: spatial locations, structural positions, locations along life-cycles, contextual characteristics, and others.

The morphological view focuses on the mereological and mereotopological structures of various categories and entities.

The flow of matters may be reflected by changing qualitative and quantitative attributes of resources and tasks along their life-cycle. With this regard the functional view centers on functional characteristics, i.e. the qualitative and quantitative properties described in functional profiles of demands, requirements, orders, products, processes, and resources.

The transformational paradigm intends to address all aspects of the dynamic reality along planning and control of manufacturing and logistics operations by a unified way of abstracting. Due to such epistemological openness, a number of important advantages can be ensured, namely:

1. Conceptual consistency of operations, manage and control processes, and applications. Therefore various systemic solutions can be facilitated in a comprehensive and homogenous way, like the synergy-based effectiveness and efficiency.
2. Different fragmentary paradigms can be supported, including SOA, multi-agency, Object-Oriented, and the ecosystemic paradigm.
3. Following the above, different technologies can be supported. Notably, the transformation paradigm does not solely depend on any of the technologies, while semantic technologies can amplify its power.
4. Holistic view of reality can be facilitated by sound and transparent representations of various interdependencies and discrepancies, reflected by relevant matchings, mappings, and fits.
5. Various temporal and spatiotemporal representations can be supported, ranging from the time-free, to time-rigid, and up to the real-time ones.
6. Various dynamic aspects of the phenomena can be conceptualized in terms of tasks transformations and resources, including some global functionalities, the homeostatic ones, which affect a large part or a whole operational environment.
7. Various manage and control methods can be supported in a natural way. E.g. with regard to the planning and control: push flows, pull flows, replenishment-driven, multi-project, hormone-based, herd self-controls, etc.
8. Various structural aspects can be effectively addressed by the spatiotemporal and mereotopological conceptualizations, including decomposability and modularity.
9. Layered conceptualizations can be supported, as well as the granularity of some elements of the operational environment.
10. Transformations can be represented in parallel in terms of different formalisms, like e.g.: algebraic, graph-theoretic, automata, operators, ontological, propositional, situational et al.

The intended application of transformational paradigm was to found a meta-ontology used to develop core and domain ontologies for the planning and control activities. The next section introduces such an ontological framework, while the section 8 considers its use for simulations and gamifications that could support qualitative assessments of systemic solutions and ecosystems, starting from the operations strategy level.

7 The TRANSFORMERS ontological framework

This section introduces a novel spatiotemporal ontology based on a mereotopological theory. Based on a theoretical reflection from the preceding sections, and applying the transformational paradigm introduced in the previous section, as well as using the insights from the review of existing ontologies presented in chapter 2, the TRANSFORMERS ontological framework is proposed below.

The main objective of the TRANSFORMERS is to aid the activities of managing and controlling operations in the systems and open environments, like: supply chains, networks, and ecosystems. Therefore it is principally a domain ontology. Despite that it fully meets the needs and requirements of practice. In particular, it is compatible

with the common systemic solutions. Hence, it can be directly used to support commercial applications. Up to now the ontology is mostly considered as a research and educational platform. Consequently, some application extensions were also made.

The TRANSFORMERS paradigm was used to conceptualize the domain, hence the abstracting in terms of the 'tasks-transformations-resources' triad was consequently applied. The developments were supplemented with a mereotopology based theory. In particular, the demand transformations from independent demand to operations, along the planning and control processes, were fully addressed in a transformational way. Demands, requirements, orders, schedules and controls are viewed as ordered collections of spatiotemporal objects (located and timed), which are linked together by the layered and granular spatiotemporal, mereotopological and functional associations. They are also aligned with the resources in a similar way.

The following inputs from the domain knowledge were considered along the development of the ontology:

- The vocabulary used by TRANSFORMERS is consistent to the maximum possible extent with the APICS Dictionary (APICS, 2013); the trading terms were derived from (Johnson, 2002) and (D'Arcy et al., 2002); the accounting terms were derived from (Moles and Terry, 1999);
- The taxonomy of planning and control approaches, methods and patterns, was developed using the concepts presented in the following publications: (Jacobs et al., 2010), (Luczak et al., 2001), (Mahoney, 1997), (Nicholas, 1998), (Österle et al., 2001), (Pinedo, 2012), (Proud, 2007), (Silver et al., 1998), (Toomey, 1996), (Vollman et al., 1998), (Wiendahl, 1995), (Lis et al., 1994), (Fernández-Rañada et al., 2000);
- The taxonomy of workflows, process flows, knowledge and data flows, together with the models of resource allocation and use, was developed by incorporating additional insights from the following publications: (Papadopoulos and Arbab, 1998), (Pinedo, 2012), (Silver et al., 1998), (Russel et al., 2005a), (Russel et al., 2005b), (Russel et al., 2006); (Man and Schiffelers, 2006);
- The taxonomies used by specific modelling languages and software products were also examined, including: BPEL4WS, BPMN, COSA, Event-driven Process Chain (EPC) as implemented using ARIS Toolset, FileNet P8 BPM Suite, FLOWer, IBM WebSphere MQ Workflow, Oracle BPEL, UML (Unified Modelling Language) as implemented using Visual Paradigm, SAP Workflow, Staffware Process Suite, Sun ONE iPlanet Integration Server, XPDL, WebSphere Integration Developer.

The TRANSFORMERS ontology provides a formal description of the whole domain knowledge, which enables consistency checking and querying throughout the demand fulfillment lifecycle. It encompasses the spatial, temporal, spatiotemporal and functional dimensions. The ontology was prototyped in the Protégé toolkit, which is equipped with the HermiT and FaCT++ reasoners. It was written in the OWL and SWRL languages. A full axiomatization and the basic required Description Logic (DL) and SPARQL (from SPARQL Protocol and RDF Query Language) queries were also developed. To support calculations, or more strictly to manipulate the data values, the SWRL built-ins were used, including the standard core SWRL built-ins, and

own built-ins developed in Java. The current status of the ontology is presented in (Strzelczak, 2015). To represent some structures of complexities that exhibit network nature, e.g. the eligible paths in a manufacturing system, the temporal RDF graphs were used, following the approach introduced in (Gutiereez et al., 2007).

The scope of operational processes encompassed within the ontology is illustrated in Fig.37. The ontology embraces all layers of the planning and control activities, including (Fig.38):

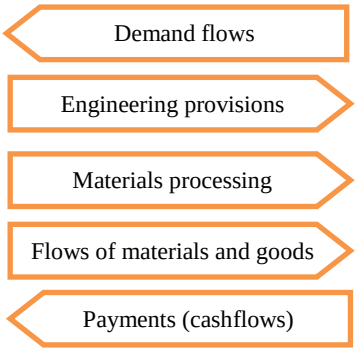


Fig. 37. A simplified view of the key operations addressed by the TRANSFORMERS ontology

- aggregate and master planning, with the Rough-Cut Capacity Planning (RCCP);
- Material Requirements Planning (MRP) and Distribution Requirements Planning (DRP), together with the Capacity Requirements Planning (CRP);
- Scheduling of orders and loads;
- Control of operations and load, including the disposal at a run-time.

The TRANSFORMERS ontology represents in a formal way all relations between tasks, transformations, and resources. Therefore all conceptualized complexities are also represented. The ontology encompasses (Fig.38):

- Meta-ontology, which provides the set of primitives based on the grounding abstractions and the founding theories, as well as the universal axioms;
- Core-ontology, which equips the whole with core vocabulary and basic axioms of the manufacturing and logistics, with regard to the planning and control activities;
- Domain ontology, which populates the core ontology with knowledge about the particular area of applications, e.g. manufacturing system;
- Application ontology, which focuses the scope of knowledge of particular appliance or application, e.g. concerning a given planning agent.

To ground the TRANSFORMERS ontology, the full set of core spatial, temporal, spatiotemporal, and functional primitives, and the required founding abstractions, have been defined and fully described in (Strzelczak, 2015). Therefore they are all but only listed in the consecutive tables. However, two entailment primitives which extend the core mereotopology, require a special attention. They are ‘Implosion’ and ‘Temporal_Implosion’.

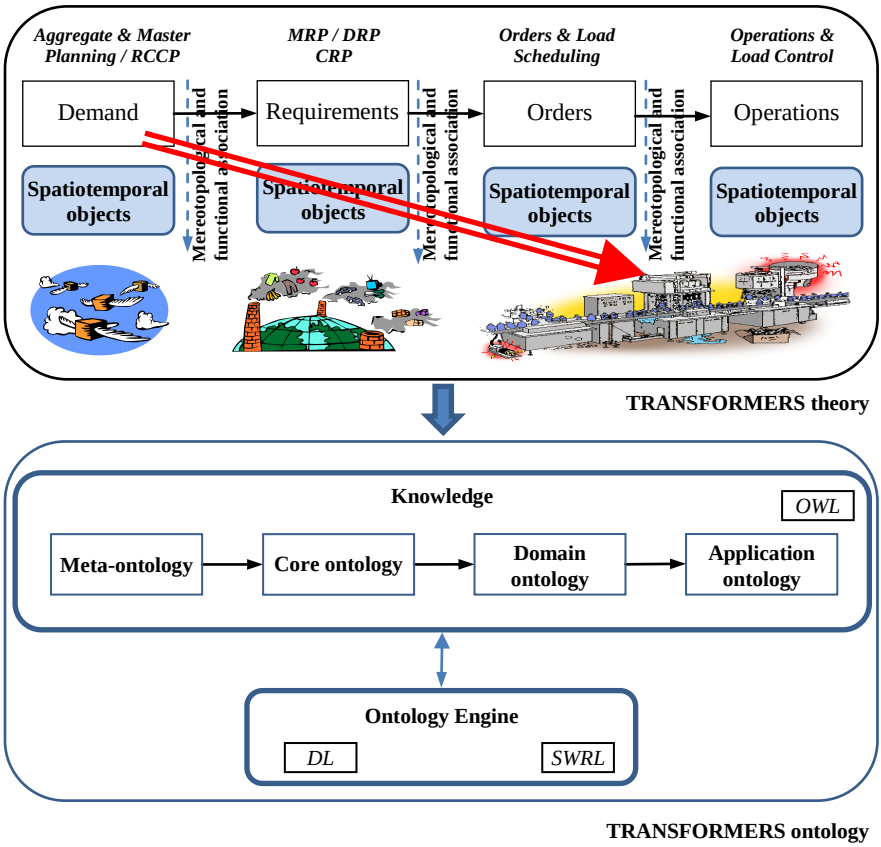


Fig. 38. The approach of TRANSFORMERS theory and ontology (case of push flows)

The first of them refers to layering. In the context of planning and controlling manufacturing and logistics operations layering may be referred to:

- Allocation and timing of independent and dependent demand;
- Allocation and timing of capacities;
- Functional associations of demands, products, technologies and capacities; they may exhibit both, spatial and spatiotemporal nature.

Altogether the required layering abstractions may be both, spatial and spatiotemporal. An example of spatial and spatiotemporal layering is exhibited when representing associations of the functional profiles of capacities at different levels of capacity planning, i.e. RCCP, CRP, and the two levels of loading. At the beginning a spatial check is done to verify the fits between demand functional profiles and capacity profiles. The second step, i.e. scheduling or loading capacities involves spatiotemporal layering. The two examples of common functionalities supported by layering and granularity are the pegging and the bottom-up planning and re-planning.

An additional challenge concerning formalization of layering and granularity takes place, when the two aspects have to be considered simultaneously, which often happens in manufacturing and logistics. Furthermore, time related layering or granularity, i.e. in reference to perdurant items, provides another difficulty, as ordered partitions have to be incorporated into the formalization.

To emphasize, the discussed issue will be even more sensitive in the future, in the open knowledge driven environments of the servitized operations (Baines et al., 2009), or in the subscription markets that tend to become a new form of economic institutions (Lev-Ram, 2014).

To illustrate the common features of layering in manufacturing and logistics, the layering of demand is taken as an example, which is normally as follows (Fig.39):

i. *Proper entailment*: The dependent demand is isomorphic with the independent demand. This happens e.g., when the lot-for-lot method of batch sizing is applied. It is important to emphasize, that the demand at a lower layer is qualitatively different, as it does not refer to the same item. Therefore the entities at different layers are not the same individuals, i.e. in the substantive and logical sense. They are normally offset, i.e. in temporal sense, and quantitatively different. Hence the core mereotopology, like the common axiomatizations of layering (Donelly, 2003; Donelly, 2004), are not relevant. It means that the commonly accepted axiomatization of the parthood and proper parthood relations are not suited herein (Leśniewski, 1992; Smith, 1996). The 'Explosion' and 'Implosion' primitives can be used to indicate the correspondence of demand, in different layers. Both terms are semantically consistent with the common terminology used within the domain of interest (APICS, 2013).

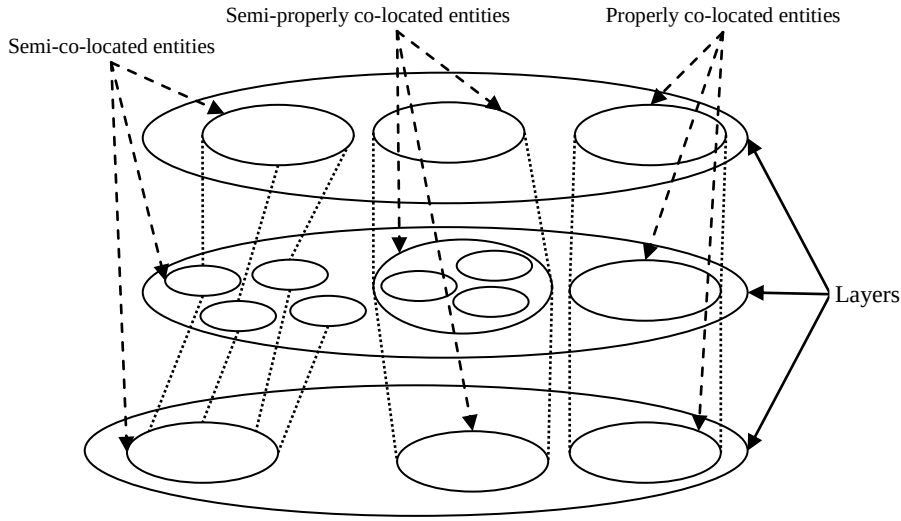


Fig. 39. Common mereotopological patterns of simultaneous layering and granularity in the manufacturing and logistics domain

ii. *Semi-proper entailment*: The dependent demand is not isomorphic with the independent demand, but is homomorphic. Split and fusion of batches provide a simple example.
iii. *Improper entailment*: The dependent and independent demand are not homomorphic, i.e. they are heteromorphic.

The above description provides a first glance view of the layered mereotopology, which is suited to domain and supports all layered associations, i.e. including the fits of demand, capacities and of functional profiles. A full description of its concepts together with presentation of the theoretical foundations is provided in (Strzelczak, 2015). In particular the absorption axioms are presented therein, which are analogs of the absorption theorems of the theory of sets. However, like it happens with the mereology, the mereotopology of layers departs from the theory of sets in its fundamental formulation given by Georg Cantor.

The above mereotopology of layering can be extended to a spatiotemporal mereotopology by the timed labelling of universals and individuals, as this approach suits to the domain and is consistent with the existing knowledge and practices. A particular advantage of this approach is the efficacy of querying, which at best can be polynomial. It results from a particular property of domain, i.e. from the typical high frequency and limited scope of ongoing changes, which normally directly affects very few individuals within the ontology (Gutierrez et al., 2007).

Table 4 lists the spatial and temporal primitives, which support the founding relations within the TRANSFORMERS meta-ontology. Considering the assumptions of theory of sets, of mereology, and of mereotopology, the spatial primitives enable definition of spatial and structural, i.e. mereological relations of tasks, transformations, and resources. The temporal primitives enable incorporation of the time dimension into the meta-ontology, therefore spatiotemporality can be addressed within it.

Table 4. Spatial and temporal primitives

Spatial primitives	Temporal primitives
Parthood	Temporal_Parthood
Proper_Parthood	Temporal_Elementhood
Overlap	Temporal_Area
Underlap	Temporal_Location
Area	Temporal_Connection
Location	Temporal_Abutement
Cover	Temporal_Containment
Containment	Temporal_Overlap
Connection	Temporal_Underlap
Abutment	Temporal_Coincidence
External_Connection	Temporal_Precedence
Coincidence	Temporal_Sequence
Explosion	Temporal_Explosion
Direct_Explosion	Temporal_Implosion
Implosion	Temporal_Direct_Explosion
Direct Implosion	Temporal_Direct_Implosion
Straddlement	Temporal_Straddlement

Table 5 presents the functional primitives. The first six of them enable to define the required functional characteristics of tasks, transformations and resources. Therefore the functional matchings and mappings are made possible between them, when indispensable. The functional entities should not be confused with the functional properties of data and objects, as applied in the OWL language. The last two primitives are used to describe condition of an ecosystem or its part in a qualitative or quantitative way.

Table 5. Functional primitives

Functional primitives
Profile_Of_Resource
Profile_Of_Task
Profile_Of_Transformation
Skill_Of
Functionality_Of
Quality_Of
State_Of
Condition_Of

The spatiotemporal primitives are vehicles to incorporate the theory of spatiotemporal mereotopology into the meta-ontology. The grounding abstractions used to define the control entities, are listed in Tables 6÷8. They are presented in details in (Strzelczak, 2015). In many cases there is no better way to represent these control primitives, than by some graph structure. With this regard the approach of temporal RDF graphs introduced in (Gutiereez et al., 2007), was adapted in the meta-ontology.

Table 6 introduces the abstractions for controlling flows of material, work, and cash. They are used along induction, explosion, deduction and aggregation of tasks. Various aspects are encompassed, including: bulking and splitting, synchronization, structuring, state-based controlling, and withdrawals.

Table 8 defines the control abstracting patterns, which represent the exchange and flows of knowledge and data. They can be applied to define various relations used to describe the external or internal knowledge and data interactions, the routing and transfer patterns, the flow triggering patterns, as well as the visibility of knowledge and data.

Table 7 lists the control abstractions that are related to the resource allocation and use. They can be used by various phases of planning and control, to load and assign resources. Considering a wide variety of possible settings for the planning and control activities, various aspects of allocating and using resources are represented in terms of these abstractions, namely: infinite and finite loading, pulling-based allocation, pushing-based allocation, detouring, auto-starting, visibility of configurable work items allocation, and finally using of multiple-resources. With regard to the future environments, which are expected to align clients and bidders of manufacturing and logistics operations, like the Production Internet, additional relevant abstractions were incorporated to equip resources with the abilities of offering and self-loading.

Table 6. Temporal mereological and mereotopological abstractions: control structures of workflows, material flows, and cashflows

Structural Patterns	Batching, Splitting and Synchronization
Arbitrary Cycle	Batch for Batch
Structured Loop	Periodic Batch
Recursion	Fixed Batch
Multiple Instances without Synchronization	Maximum Batch
Multiple Instances with a priori Design Time Knowledge	Minimum Batch
Multiple Instances with a priori Run Time Knowledge	Calculated Batch
Multiple Instances without a priori Run Time Knowledge	Sequence
Triggering and State-based Patterns	Semi-parallel Split
Deferred Choice	Parallel Split
Transient Trigger	Synchronization
Persistent Trigger	Exclusive Choice
Interleaved Parallel Routing	Multi-Choice
Critical Path	Acyclic Synchronizing Merge
Interleaved Routing	General Synchronizing Merge
Milestone	Structured Synchronizing Merge
Cancellations	Simple Merge
Implicit Termination	Multi-Merge
Explicit Termination	Structured Discriminator
Cancel Demand	Blocking Discriminator
Cancel Requirement	Cancelling Discriminator
Cancel Order	Generalized And-Join
Cancel Operation	Structured Partial Join
Cancel Operation Task	Blocking Partial Join
Cancel Multiple Instance Item	Static Partial Join for Multiple Instances
Finish Multiple Instance Item	Dynamic Partial Join for Multiple Instances
	Cancel Partial Join
	Cancel Partial Join for Multiple Instances
	Thread Merge
	Thread Split

The abstractions introduced in Tables 6÷8 were developed aiming at three important objectives. Firstly, they can be used, more or less directly, to define the content of the meta-ontology and the core ontology, including the primitives. With this regard a well done development of an ontology may result in improved intelligibility, simplicity, and therefore efficacy of the ontology. Secondly, the abstractions provide altogether a vast basis for definition of the dynamic complexities, that arise along manufacturing and logistics operations and their planning and control. In particular, checking of all relevant fits that are exhibited in Fig.21÷24, is made possible. Consequently, a well done conceptualization of the complexities can effectively and efficiently support the novel systemic solutions and functionalities. Finally, the abstractions provide a comprehensive set of theoretical means to drive composition and decomposition of tasks and transformations as Web services. This applies downward to the level of operations control.

Table 7. Temporal mereological and mereotopological abstractions: control structures for allocation and use of resources

Allocation Primitives	**Push Primitives**
Direct Allocation	Offering –Single Resource
Capability-based Allocation	Offering – Multiple Resources
Load-based Allocation	Allocation - Single Resource
Cost-based Allocation	Random Allocation
History-based Allocation	Round Robin Allocation
Dependability-based Allocation	Shortest Queue
Reliability-based Allocation	Quickest Queue
Role-based Allocation	Early Distribution
Deferred Allocation	Distribution on Enablement
Disposal	Late Distribution
Authorization	**Detour Primitives**
Separation of Duties	Delegation
Case Handling	Escalation
Retain Familiar	Deallocation
Organizational Allocation	Stateful Reallocation
Automatic Execution	Stateless Reallocation
Auto-Start Primitives	Suspension
Commencement on Creation	Resumption
Commencement on Allocation	Skip
Repetitive Execution	Pre-Do
Piled Execution	Redo
Chained Execution	Rework
Pull Primitives	**Visibility Primitives**
Resource-Initiated Allocation	Configurable Unallocated Item Visibility
Resource-Initiated Execution – Allocated Item	Configurable Allocated Item Visibility
Resource-Initiated Execution – Offered Item	**Multiple Resource Use Primitives**
System-Determined Queueing	Simultaneous Execution
Resource-Determined Queueing	Additional Resources
Autonomic Selection	External Resources (Outsourcing)

The meta-ontology provides all capacities needed to represent the entities and their relationships, considering the four dimensions, i.e.: functional, spatial, temporal and spatiotemporal. It must be underlined, that the primitives and abstractions presented in Tables 4÷8 do not have to be directly mirrored within the ontology, i.e. in terms of classes, objects or properties. They should be rather used in a functional way, i.e. as adaptable multi-purpose concepts for definition of the meta-ontology, and of the core ontology. This actually takes place, and the founding primitives and abstractions are normally built in the items of the ontology in a hidden way. Otherwise, the ontology would lost a lot of its transparency, intelligibility, and also of efficacy.

The top-level classes, as well as the top object and data properties were categorized according to the transformational paradigm, i.e. as task, transformation and resource super-categories. As the ontology currently incorporates over ten thousand of classes, properties, axioms and queries, the bird-eye view of the ontology is presented in this book, i.e. by selected sub-categories, which are exhibited in the following diagrams.

Table 8. Temporal mereological and mereotopological abstractions: control structures of knowledge and data (K&D) flows

K&D Visibility	External K&D Interactions
Task Visibility	Pushed Interaction - Task to Outside
Block Visibility	Pulled Interaction - Outside to Task
Scope Visibility	Pushed Interaction - Outside to Task
Multiple Instance Visibility	Pulled Interaction - Task to Outside
Case Visibility	Pushed Interaction - Case to Outside
Workflow Visibility	Pulled Interaction - Outside to Case
Outside Visibility	Pushed Interaction - Outside to Case
Internal K&D Interactions	Pulled Interaction - Case to Outside
Interaction between Tasks	Pushed Interaction - Workflow to Outside
Interaction - Block Task to Sub-workflow Decomposition	Pulled Interaction - Outside to Workflow
Interaction - Sub-workflow Decomposition to Block Task	Pushed Interaction - Outside to Workflow
Interaction - to Multiple Instance Task	Pulled Interaction - Workflow to Outside
Interaction - from Multiple Instance Task	Pushed Interaction – Material-flow to Outside
Interaction - Case to Case	Pulled Interaction - Outside to Material-flow
	Pushed Interaction - Outside to Material-flow
	Pulled Interaction – Material-flow to Outside
K&D Transfers	**K&D Routings**
Passing by Value – Incoming	Task Precondition - Existence of K&D
Passing by Value – Outgoing	Task Precondition - Value of K&D
Passing - Copy In/Copy Out	Task Postcondition - Existence of K&D
Passing by Reference -Unlocked	Task Postcondition - Value of K&D
Passing by Reference - Locked	Event-based Trigger
Transformation - Input	K&D-based Trigger
Transformation - Output	K&D-based Routing

Fig.40 present the top subclasses of tasks. All levels of demand processing are covered, i.e. from the demands to the operations disposals. Apart of that the performance targets are incorporated, i.e. KPIs (Key Performance Indicators), which refer to the manufacturing and delivery tasks (Fig.41), as well as those tasks, that are related to the planning of other tasks.

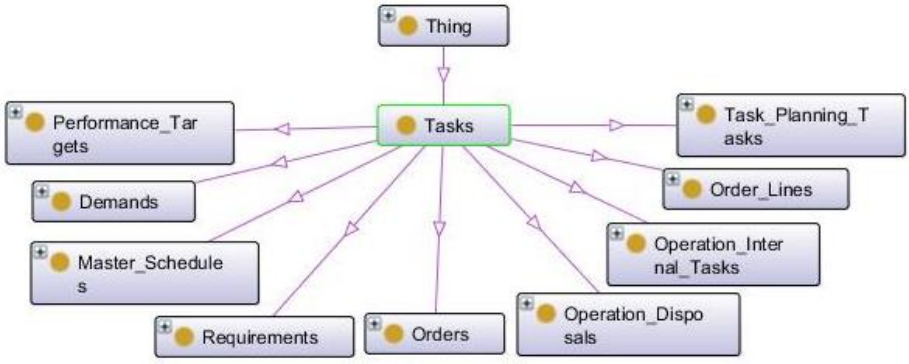


Fig. 40. Top sub-classes of tasks

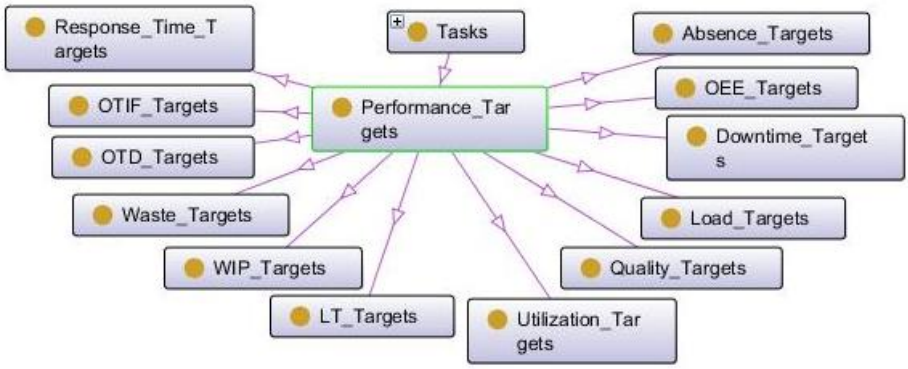


Fig. 41. Sub-classes of performance targets

Fig.42 and 43 exhibit the subclasses of requirements and orders.
The 'requirements' class incorporates three types:

- the demand rooted requirements for products, subassemblies, fabricated parts and materials; these are expressed in two ways, i.e. by schedules, and by maximum rates, like e.g. in the 3C system;
- the requirements for engineering works (designs, technologies, etc.);
- the capacity requirements for manufacturing resources.

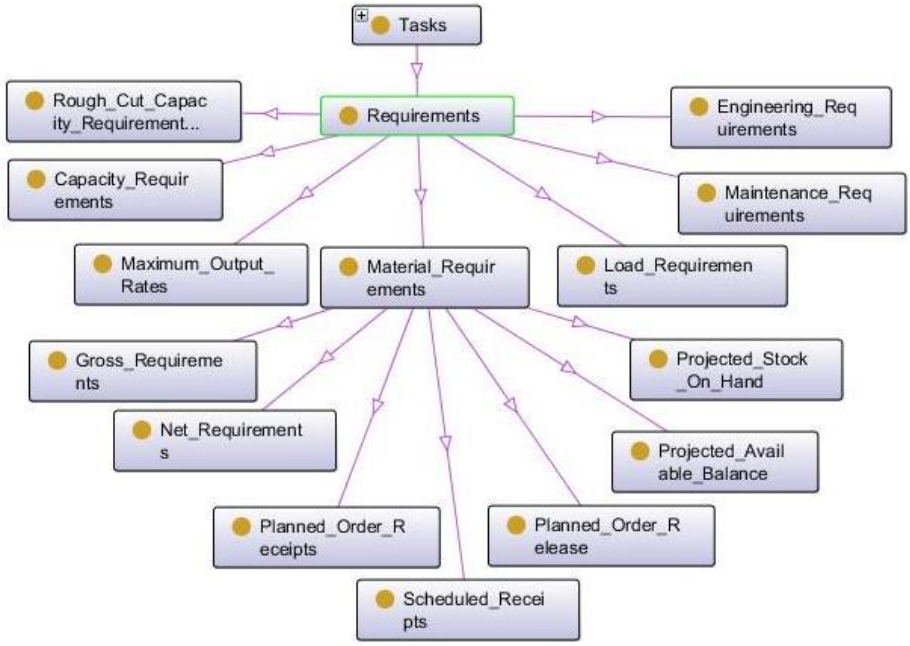


Fig. 42. Sub-classes of requirements

The ‘orders’ class incorporates all addressed processes, e.g. financial orders are represented. To illustrate the properties of tasks, Fig.44 lists the object properties of manufacturing orders, and similarly Fig.45 presents the data properties.

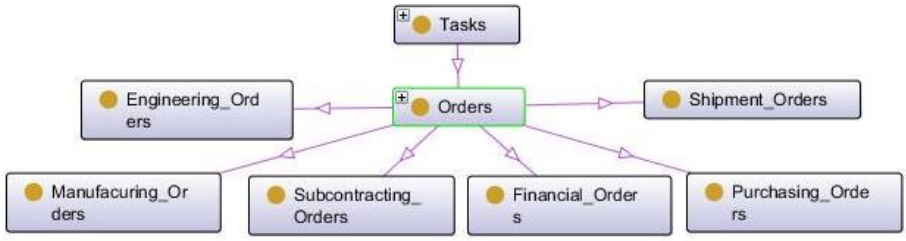


Fig. 43. Top sub-classes of orders

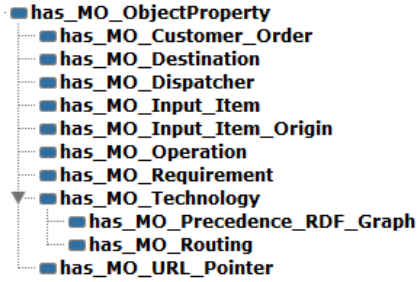


Fig. 44. Manufacturing order object properties

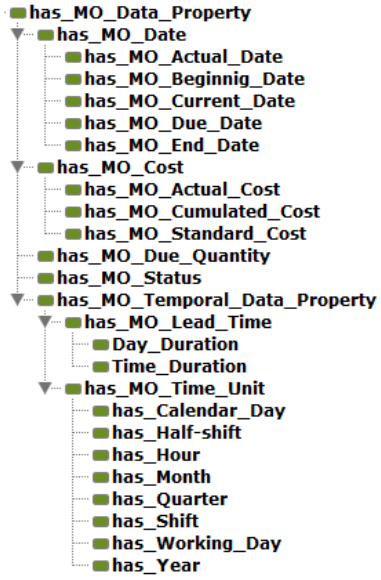


Fig. 45. Manufacturing order data properties

The super-class of resources was only selectively developed due to the availability of the P-MSO manufacturing systems ontology, which was developed at Politecnico di Milano by the team of Prof. Marco Garetti, and refined along the works of eScop project. The P-MSO ontology provides a vast taxonomy of manufacturing and logistics equipment. Therefore the TRANFORMERS ontology assumes the use of P-MSO within it. Nevertheless, other classes of resources were defined.

The class of resources includes the sub-class of environments, to enable representation of inter-organizational structures, economic institutions and ecosystems.

Other classes incorporate people, manufacturing and logistics resources, control hardware, software resources, materials, financial resources, knowledge, and finally own systems (Fig.46). The sub-class of software resources includes agents (Fig.47). This class enables to support the multi-agent settings by the means of ontology, including the agent-based ontology-aided simulations and gamifications. This part of TRANSFORMERS ontology was designed with the aim to support possible non-hierarchical planning and control structures for manufacturing and logistics operations, e.g. auction based control, peer-to-peer bidding and similar. Therefore the ontology is preset to aid the novel systemic solutions, like those discussed in chapter 2.

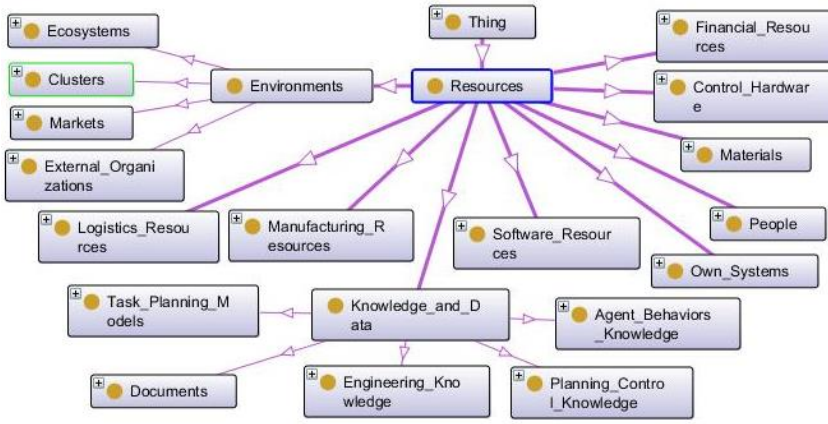


Fig. 46. Sub-classes of resources

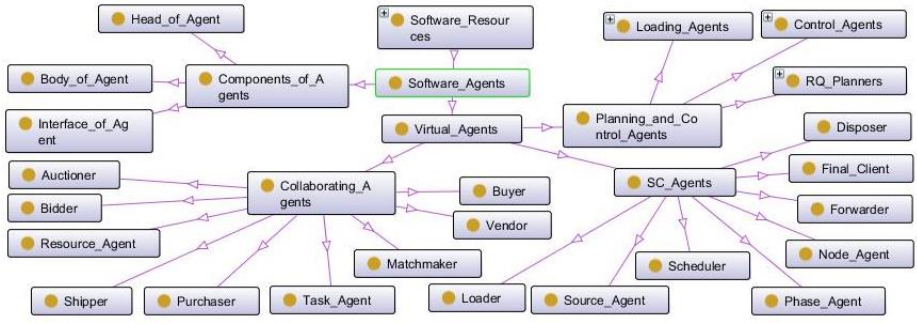


Fig. 47. Sub-classes of agents

An important part of the ontology is knowledge. The components of knowledge enable planning and control of operations, and the run of operations. Knowledge universals are used to create individuals along the operations performance. The classes of planning and control knowledge are depicted in Fig.48. All levels of planning and control are considered, like the all major approaches, including: the MRP approach, replenishment and Kanbans, 3C, the common types of batching, leveling, Input-Output control, and prioritizing. Priority rules are exhibited separately in Fig.49.

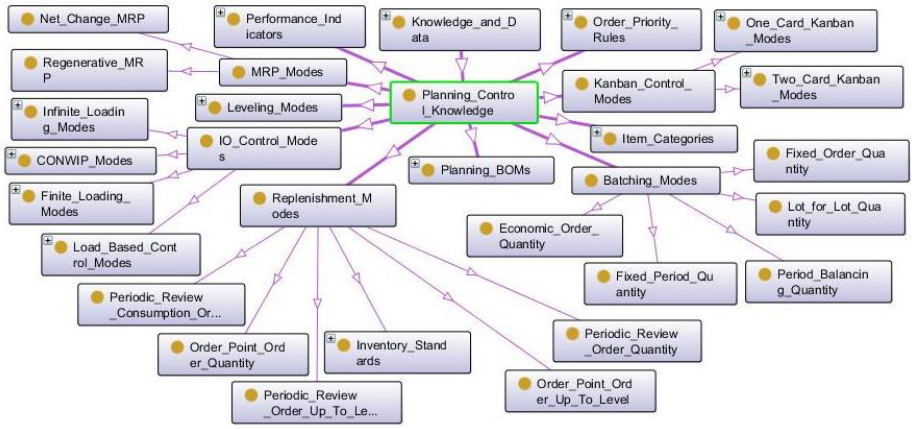


Fig. 48. Sub-classes of planning and control knowledge

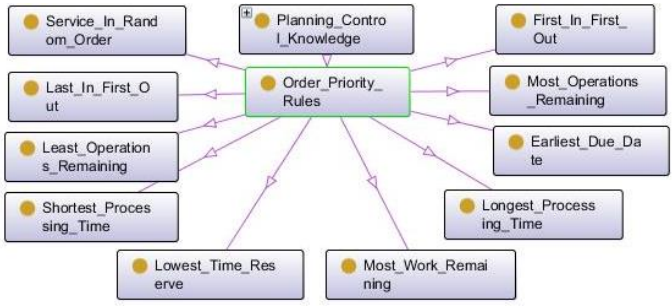


Fig. 49. Sub-classes of priority rules

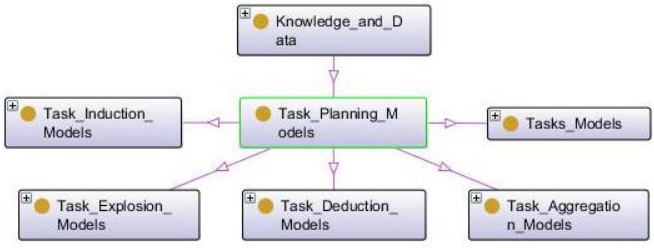


Fig. 50. Sub-classes of tasks planning models

Fig.50 provides a view of the classes of task planning models, which are used in a run-time mode along composition and decomposition of tasks. The classes reflect the basic modes of planning the tasks, which were discussed in the preceding sections.

Fig.51 introduces the structure of engineering knowledge. The two major parts of knowledge addressed herein are representation of products, i.e. in terms of the engineering BOMs (Bill-of-Materials), and the knowledge on technologies.

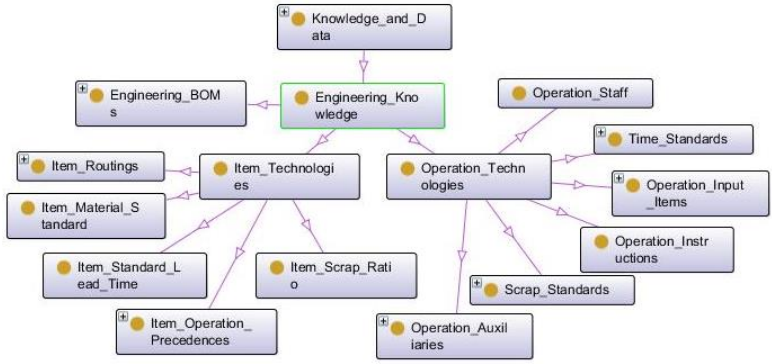


Fig. 51. Classes of engineering knowledge

The classes of transformations are represented in the ontology in a twofold way (Fig.52), i.e. by the current transformations, and the past ones. Historical records can be used for analytical reasoning purposes. Apart of the resources and tasks transformations, the current status is represented by a separate class of 'states-configurations-locations'. This facilitates triggering of tasks and transformations, which are conditioned by the state of matters (e.g. of the whole ecosystem). In particular, variability related states are represented, which can be referenced to various items. This class provides capacities to monitor variability and turbulent behavior of processes, systems and ecosystems. Therefore ontology-aided simulations and gamifications are made possible, aiming at ex ante discovery of the unwelcome behavioral properties.

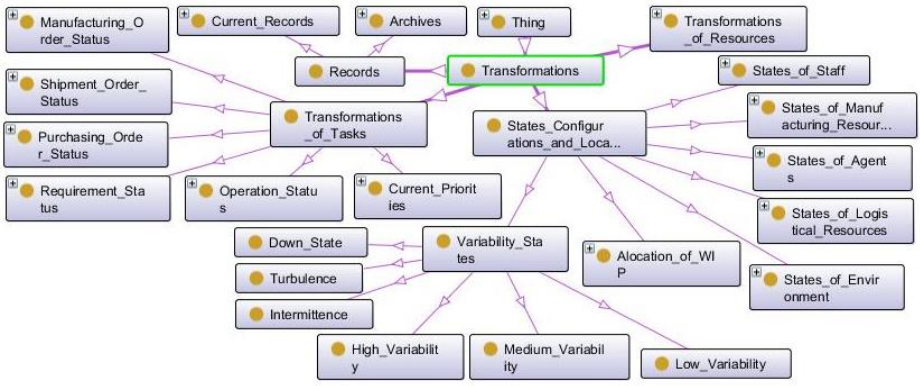


Fig. 52. Top classes of transformations

The subclasses of resources' transformations of are exhibited in Fig.52. These include reconfigurations and relocations, setups, as well as all other transactions that result in direct changes of resources.

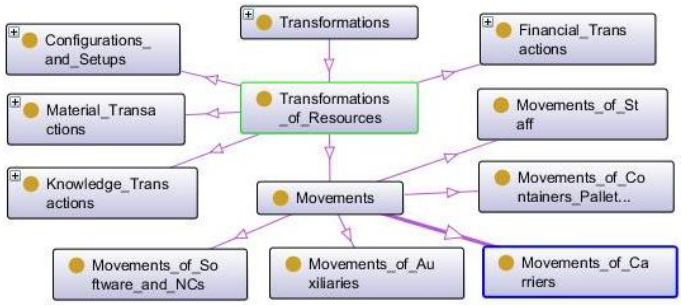


Fig. 53. Top sub-classes of transformation of resources

The development of TRANSFORMERS ontology is still ongoing. Up to now it was twice refined. The base of rules and the reasoning layer are being gradually extended, using the Description Logic (DL) and the SWRL (Semantic Web Rule Language) languages.

The ontology was developed with an excess. One objective for that is to ensure the possibility of further extensions at the core and domain level, with regard to the requirements of other subdomains of manufacturing and logistics. The second objective is to support novel systemic solutions that might arise in the future. With this regard a wide range of universals was incorporated, like planning and control patterns, hierarchies, operational structures, including non-hierarchical, some arising forms of economic institutions, and so on. The excessive capacities can be easily reduced and tailored to particular requirements of given domain and application domain.

Finally, the ontology was intentionally suited to the research purposes, especially with regard to qualitative modelling of operational environments, like networks, supply chains, and ecosystems. This way behavioral characteristics of various environments and managerial settings, can be researched. A prototype use case of that kind is presented in the next section, which focuses on the use of ontology for the simulation and gamification purposes.

It is expected that from the beginning of 2016 the TRANSFORMERS ontology will be freely accessible to the public for the research and educational purposes.

8 Ontology-aided simulation and gamification

Although the TRANSFORMERS paradigm and ontological framework was primarily designed to aid the decision making and automation along operations planning and control, it can be also used for purely analytical purposes. The potential functionalities of that kind incorporate simulation and gamification, which can be used for purely research purposes (Fig.54), or alternatively to aid the strategic or tactical decisions.

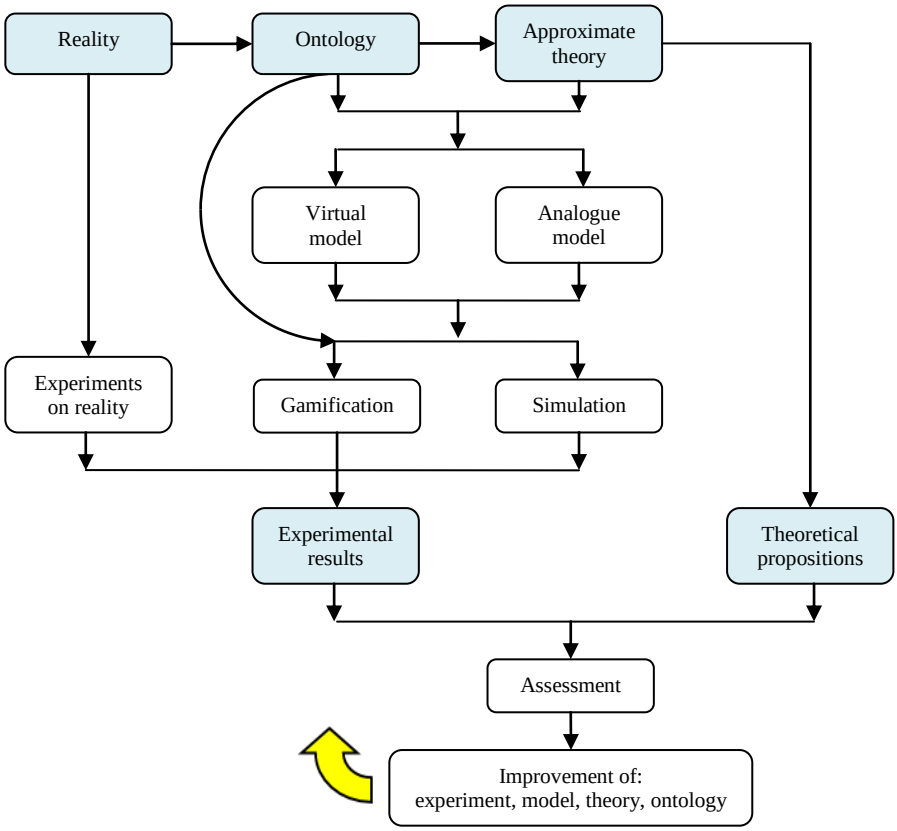


Fig. 54. Epistemological positioning of simulation and gamification the research-life-cycle view

Simulation relies on using a model or an ontology to imitate behavior of particular aspects or parts of reality. Simulation models may be virtual or analogue.

Gamification involves people (gamers) or their virtual analogues, i.e. agents, following the concept of Agent Based Modelling (ABM) (Macal and North, 2006)[6].

Both, simulations and gamifications, are developed with regard to some ontological and theoretical propositions. Simulations and gamifications can be ontology–aided. Analogue models may be physical objects, chemical reactors, and bio- or eco-reactors (Strzelczak, 2013).

Virtual and analogue models can be hybridized, and additionally robotized (Strzelczak, 2013). The latter means that an assigned ontology-aided agent plans and supervises experiments[7].

[6] ABM should not be confused with the Multi-Agent-Systems (MAS).

[7] Robot can be also used to plan and supervise a sole computer simulation, as well as to experiment on reality.

Table 9 reviews the existing simulation methodologies, to enable their comparison with the proposed approach. Two basic types are discrete and continuous simulation. A hybrid approach was proposed by Umeda (2008) for supply chain management analysis, and the Hybrid Chi for manufacturing systems (Man and Schiffelers, 2006).

Table 9. Comparison of basic classic simulation methodologies[8]

Formalism	Key features
Discrete simulation	
Discrete Event	A queue of events is maintained and ordered by the time they should occur. The processed events trigger new events. Time advances from one event to the next.
State/Action	The reality is represented by a series of discrete states, whereby each state depends on a previous one. A state has an associated action and an event that will cause that state to change to another. The transition from one state to the next is not sequential; each state can lead to any other state. System responds to events by transition to other states.
Network Modelling	Flows between the nodes of a network are modelled considering the network topology and predefined processes.
Continuous simulation	
Continuous	Continuous real functions are used to represent a continuously changing system. Model continuously tracks responses according to a set of equations, which are typically differential equations.
Continuous with discretized time	Time advances in equal increments, and functional values change based directly on changes in time. Aggregate interactions are considered in a network of interconnected feedback structures.
Hybrid simulation (continuous and discrete)	
Hybrid Chi	Applies the Hybrid Chi processual algebra to represent the ongoing and mixing discrete and continuous processes.
Discrete Rate	Processes are attributed by rates, other parameters, and events. Rates and other values are recalculated whenever some type of event occurs.

It is proposed herein to apply ontologies to implement qualitative simulation and gamification. The term ‘qualitative’ is employed with a particular and exact meaning, which departs from the common interpretation of this word. Qualitative means that a combination of particular features is available along simulation and gamification, i.e.:

- Reality is represented by the ontology, which can be eventually supplemented with a quantitative representation; in the latter case qualitative representation is superordinate to quantitative representation; ontology is basically a qualitative mean;
- Dynamic behavior is represented by flows of tasks, transformations, and changing resources, i.e. in a nonlinear phenomenological way, which means that responses to the ongoing changes do not need to be conditioned by a single variable;

[8] Simulation and gamification methodologies used by other domains of interest are not considered herein.

- Knowledge embedded in the ontology is extendable by gathering experience, and by pre-set cognitive capacities of reasoning and learning built within the ontology;
- Autonomous agents, who may exhibit emotional, herd and intelligent behavior, can be incorporated, who in the case of gamifications can be humans.

The qualitative simulation is entirely different from the existing simulation approaches. It generalizes them. It enables anticipation and assessment of behavioral and emergent qualities of systems and ecosystems. In particular, irregular behaviors can be investigated, like the turbulent or chaotic ones.

The simplified view of functional architecture for ontology-aided simulation is depicted in Fig.55. The architecture of ontology-aided gamification is similar. If the agent is human, she interacts along the game through her user interface. The universality and flexibility of the eScop architectural approach is fully exploited by the simulation architecture, which clearly exhibits a strong similarity to the eScop structure of the three architectural layers. The physical layer is not directly represented for natural reasons. It is subject of simulation, and it is virtually represented in a reduced way within the ontology, according to the presumptions. The exception is gamers who are involved in the gamification.

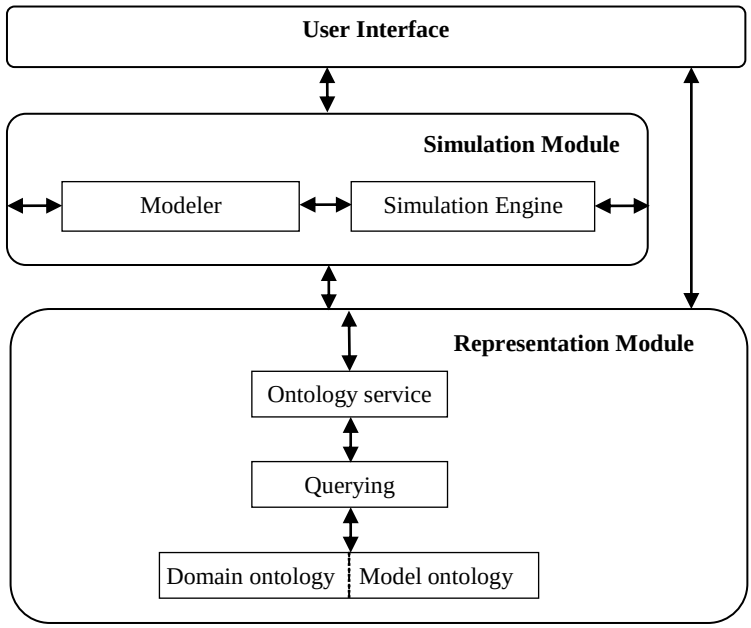


Fig. 55. Functional architecture of ontology-aided simulation

The scenarios of gamification and simulation are presented as the UML sequence diagrams, in Fig.56 and Fig.57 respectively. The major difference between the two scenarios is in the additional lifelines, which exist in the gamification scenario. They are the lifelines of gaming agents.

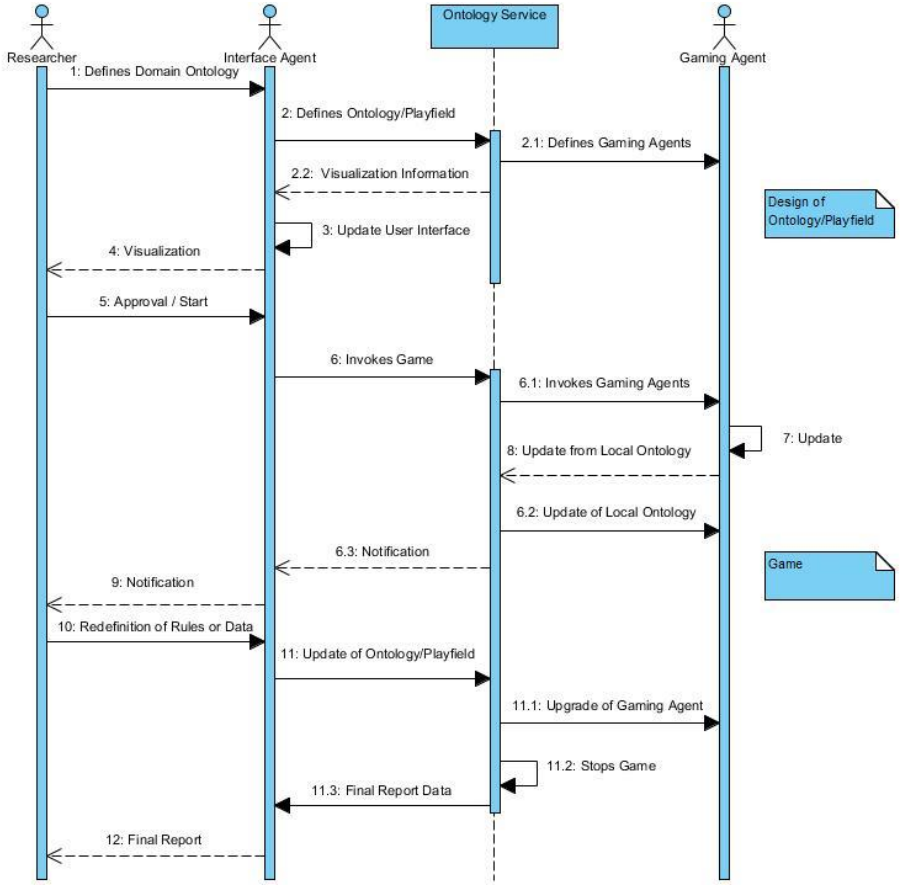


Fig. 56. UML sequence diagram for ontology-aided gamification

Fig.58 depicts the basic cooperation primitives, which define the semantics for task activation and execution by interacting agents. The actual set of cooperation primitives is normally larger, and it has to be adapted to the pragmatics of cooperation and executions of tasks within a given subdomain.

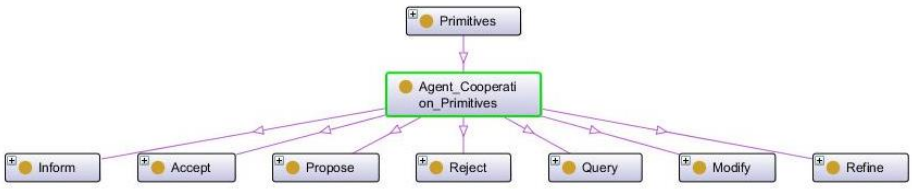


Fig. 57. Cooperation primitives of interacting agents

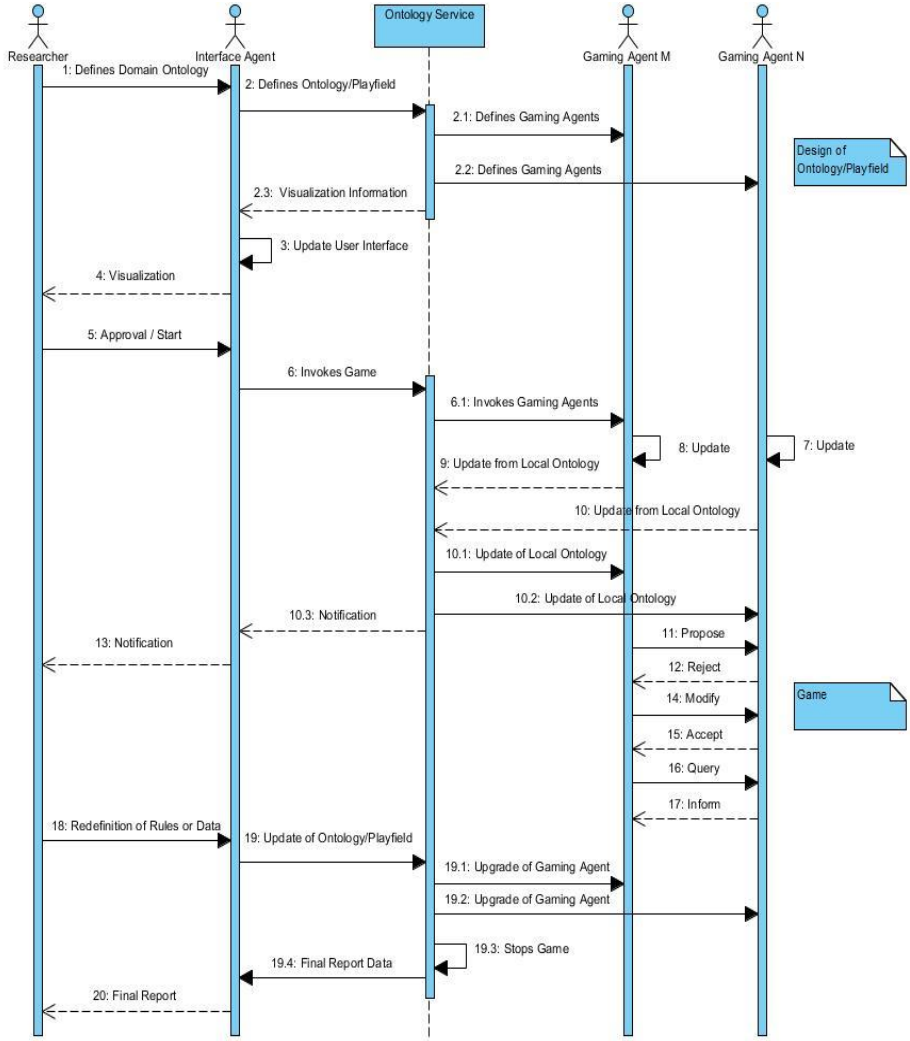


Fig. 58. UML sequence diagram for ontology-aided simulation

The presented concept for qualitative simulation and gamification was prototyped using the available technologies. The technology view of the architecture is presented in Fig.58. Six ways of interacting with the ontology are applied:

i. SPARQL and DL querying to analyze and update the ontology using a standard functionality of Protégé;
ii. messaging from ontology in JSON format using a standard Protégé functionality;
iii. updating ontology in a textual mode (OWL/RDF);
iv. querying ontology using Google Apps and standard browser;
v. human interfacing using a standard functionality of Protégé;
vi. reasoning on the ontology using the HermiT.

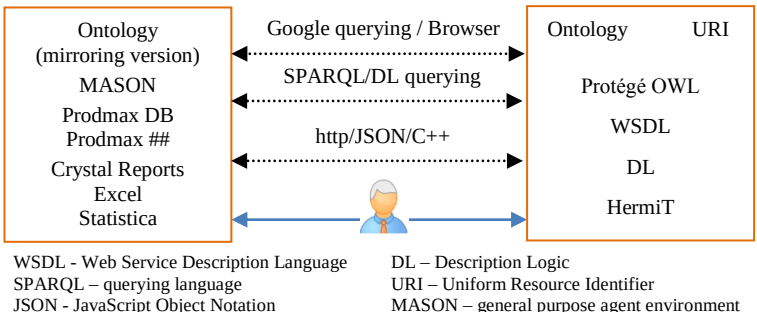


Fig. 59. Technology architecture of ontology-aided simulation and gamification prototype

Additionally a dual version of ontology is handled to enable utilization of some functionalities that are offered by dedicated platforms, namely:

- MASON to handle multi-agency;
- Prodmax DB to store all records;
- Prodmax library of planning and control operations;
- Prodmax import-export functionality;
- Presentation and analytical functionalities offered, by: Crystal Reports, Statistica and MS Excel.

Additionally the Berkeley Madonna package is used to support the simulations of continuous phenomena, which are represented by the differential equations. It is done by merging the main simulation, which is operated by the dual ontology, with the calls to the Berkeley Madonna runs.

The scope of targeted uses of gamifications and simulations is presented as the UML Use Case diagrams, in Fig.60 and Fig.61 respectively. As yet the gamifications and simulations of manufacturing and logistics operations were centered around two particular aspects: the behavior of decision makers, and other impacts of complexity.

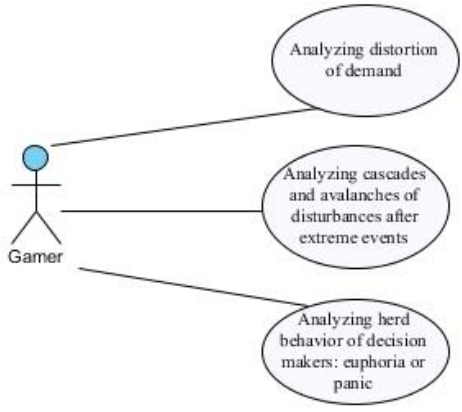


Fig. 60. Use Case diagram of gamifications

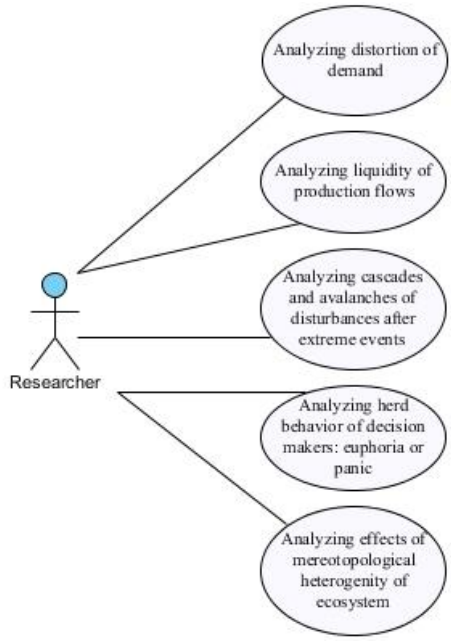


Fig. 61. Use Case diagram of simulations

Networked environments of intelligent agents may exhibit paradoxical behaviors. Examples were provided by Braess (1964) and Cohen-Kelly (1990). In flow networks, adding routes and capacity to an existing network, may decrease the average flow time, or oppositely, i.e. deleting routes and capacity may improve the performance. The conditioning factors for paradoxical network behavior herein are the intelligent selfish behavior, and the level of congestion.

Another type of paradoxical behavior is represented by the so called bullwhip effect in supply chains (Lee et al., 1997). In this case the idiosyncrasy is linked to the distortion of demand information, and to the asymmetric perception of risk by agents. All above examples illustrate the unforeseenability of behavior manifested by the complex environments of networked intelligent agents. There are very few methods to research behavioral qualities of such environments, and the qualitative simulation or gamification are among them, providing sound and robust, while not resourceful aid. Consequently the domain and application parts of the TRANSFORMERS ontology were prototyped to enable qualitative simulation of supply chains and ecosystems.

With regard to the domain of interest, the decision making agents are responsible for planning and ordering manufacturing and logistics operation. The purpose of simulation is to investigate the possible idiosyncratic behavior of flows, and in particular to enable anticipation of cascades and avalanches of operational continuity disruptions. Following (Strzelczak, 2012) the following conditioning factors are considered:

1. The level of completeness and connectivity of the manufacturing and logistics networks, i.e. the mereotopological characteristics;

2. The diversity of operational characteristics of systems and environments, which itself may be a factor of turbulences or disruptions, of the operational processes and of the states of systems or ecosystems;
3. The systemic solutions concerning the demand processing;
4. The herd behavior of clients and decision makers, which accelerates or tames the demand, due to the occasional asymmetry of the demand risk perception;
5. The extreme events, which affect some nodes of production network, or the ecosystem as a whole.

The behavior of decision makers, i.e. the agents who decide about requirements, including internal requirements and purchasing requirements, is subjected to various impacts, which according to the theory of prospect may result in asymmetric perceptions of the decision factors, including the risk (Kahneman and Tversky, 2008). Among the possible phenomena with this regard the following two can be listed:

- Panic against shortage, or alternatively market euphoria, and so on; this kind of emotions can diffuse between agents, or directly to the agents from the environment; to represent the spread of panic in the peer-to-peer mode, the Bass model was adapted (Bass, 1969); to represent the impact from environment, the model proposed by DeGroot (1974) was adapted; to represent opinion forming, the Axelrod model was adopted (Axelrod, 1997);
- Learning from experience, e.g. by observing vendors or communicating with them, or by observing the environment; this can be implemented by the Bayesian (observational) learning; this way a primitive cognitive capacity can be built into the simulation or gamification model.

The research performed using ontology-aided multi-agent simulations enabled by an experimental evidence to discover the impact of factors on the behavior of operational environments. They are provided in a separate publication, which is now in printing (in Polish). Among the most significant factors there are:

- network topology: of flows, and of technical and organizational tiers,
- behavioral determinants: asymmetric perception of demand during purchasing decisions; operational determinants: operational liquidity; operational policies; profile operational risks;
- the level of completeness and connectivity of the networks, which is related to the freedom of choosing alternative vendors or distribution channels,
- the diversity of operational environments within the network, which is related to diversification of operational characteristics of different segments of the network.

The performed ontology-aided experiments provided the first proof of concept of the transformational approach, and of relevance of the TRASNFORMERS ontology.

The distinctive advantage of ontology-aided simulation or gamification is due to the use of almost the same ontology for experimental and operational purposes. Therefore the effort into development of simulation or gamification, which is normally a part of the operations strategy process, can be retained later on along the applications development, and then along the exploitation.

9 Summary

The discussion in this chapter addresses gaps, which were identified in reference to the ontology-aided management of manufacturing and logistics. It was argued that theoretical foundations and abstractions behind ontology languages should consider particular functional requirements:

1) Transformational perspectives on resources and tasks should be considered, including the spatiotemporal mereotopological aspects;
2) All management and control know-how should be incorporated into the ontologies, starting from the meta-level;
3) Heterarchical structures of resources, tasks, plans and controls, should be considered within the ontologies;
4) All common interdependencies and discrepancies should be addressed, as well as other complexities; various temporal fits between resources, tasks, operations, loads should be considered, including functional, structural, location (spatial) and sequential fits;
5) Novel structures of systems and processes should be supported by the developed ontologies, to meet the existing and future requirements, especially with regard to the novel forms of economic institutions.

The identified requirements define directions for further technological research, particularly in reference to:

6) Refinements and extensions of the existing ontology languages by consideration of the mereotopological and spatiotemporal aspects, which cannot be properly represented by the distributive classes and subsumption relations;
7) The theoretical weaknesses of the first order and description logic should be tamed to the possible extent, as the domain of manufacturing and logistics is subjected to non-linearity and non-monotonicity.

The proposed transformational paradigm, supplemented with the TRANSFORMERS ontology, provide together the theoretical and conceptualization means, which help to reduce many of the identified shortcomings of the existing ontological developments.

The presented approach to simulation and gamification provides capacity to anticipate the unexpected behaviors of complex systems and environments. Supply chains, production networks and operational ecosystems, are good examples of that kind.

Ontologies are self-extendable. It is possible to equip them with some simple abilities to extend the knowledge or even self-learn. Therefore some kind of simple emergent self-sufficient distributed smartness or intelligence can arise, actually in an alienated mode, not easy to be tracked or understood. This aspect of the openness of ontologies is particularly interesting. It is the first time in the history of human kind that people are able to install a limited distributed intelligence, which is not easy to be followed, and eventually may get out of the control. Examples of that kind are exhibited by the intelligent anti-terrorist systems developed using the Cyc-based ontologies.

Acknowledgement: The research work presented in this chapter has been co-funded by the Grant from Warsaw University of Technology (PSP 504/01518/1103/40.000105).

References

Adamék, J., Herrlich, H., & Strecker, G.E. (2004). Abstract and Concrete Categories - The Joy of Cats. *http://katmat.math.uni-bremen.de/acc*

Allen, J.F. (1983). Maintaining knowledge about temporal intervals. *Communications of ACM*, Vol.26/11: 832‑843.

APICS (2013). APICS Dictionary (14th edition). Chicago: APICS, 2013.

Aβmann, U., Zschaler, S., & Wagner, G. (2006). Ontologies, Metamodels and the Model-Driven Paradigm, In: C. Calero, F. Ruiz, M. Piattini (Eds.), *Ontologies for Software Engineering and Software Technology*, Springer: 249-274.

Axelrod R. (1997). The dissemination of culture: a model with local convergence and global polarization. *Journal of Conflict Resolution*, 41/3: 203-226.

Baines, T.S., Lightfoot, H.W., Benedettini, O., & Kay, J.M. (2009). The servitization of manufacturing: A review of literature and reflection on future challenges, *Journal of Manufacturing Technology Management*, Vol.20/5: 547 – 567.

Ballot, E., & Fontane, F. (2008). Rendement et efficience du transport: un nouvel indicateur de performance. *Revue Française de Gestion Industrielle*, Vol. 27/1: 41-55.

Bartalos, P., & Bieliková, M. (2011). Automatic Dynamic Web Service Composition: A Survey and Problem Formalization. *Computing and Informatics*, Vol. 30: 793–827.

Bass, F.M. (1969). A New Product Growth Model for Consumer Durables. Management Science, Vol.15/No.2: 215-227.

Berners-Lee, T., Hendler, T., & Lassila, O. (2001). The Semantic Web, *Scientific American*, 2001/V: 34–43.

Bézivin, J. (2005). On the unification power of models, *Software and Systems Modeling*, Vol. 4/2: 171-188.

Bhatt, M. (2012). Reasoning about Space, Actions and Change: A Paradigm for Applications of Spatial Reasoning. In: S.M. Hazarika (Ed.) *Qualitative Spatio-Temporal Reasoning: Trends and Future Directions*. Hershey: IGI Global: 284-320.

Bittner, T., & Smith, B. (2003a). A Theory of Granular Partitions. In: M. Duckham, M.F. Goodchild, M.F. Worboys (Eds.), *Foundations of Geographic Information Science*. London: Taylor & Francis, 2003: 117–151.

Bittner, T., & Smith, B. (2003b). Granular Spatio-Temporal Ontologies. *Proceedings of Symposium of the American Association for Artificial Intelligence* (2003): 12-17.

Brachman, R., & Levesque, H.J. (1985). Readings in Knowledge Representation. Morgan Kaufmann: pp. XVI–XVII.

Braess, D. (1968). Über ein Paradoxien aus der Verkehrsplanung. *Unternehmensforschung*, Vol.12/3: 258–268.

Casati, R., & Varzi, A.C. (1999). Parts and Places: The Structures of Spatial Representation, Cambridge, MIT Press 1999.

Chandrasekaran, R., Josephson, J.R., & Benjamins, V.R. (1999). What are ontologies, and why do we need them?, *IEEE Transactions on Intelligent Systems*, 1999/January-February: 20-26.

Cohen, J.E., & Kelly, F.P. (1990). A paradox of congestion in a queuing network. *Journal of Applied Probability*, Vol.27/3: 730–734.

Cohn, A.G., & Hazarika, S.M. (2001). Continuous Transitions in Mereotopology. In: *Proceedings of the 5th Symposium on Logical Formalizations of Commonsense Reasoning (Commonsense-2001)*: 1-10.

Cohn, A.G., & Varzi, A.C. (1998). Connection Relations in Mereotopology. In: H. Prade (Ed.), *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI'98)*: 150-154.

Colombo, A.W., Karnouskos, S., Mendes, J.M., & Leitão, P. (2015). Industrial Agents in the Era of Service-Oriented Architectures and Cloud-Based Industrial Infrastructures. In: P. Leitão and S. Karnouskos (Eds.) *Industrial Agents: Emerging Applications of Software Agents in Industry*, Elsevier: 67-87.

Clarke, B.L. (1981). A calculus of individuals based on 'connection'. *Notre Dame Journal of Formal Logic*, Vol. 32/3: 204-218.

Crainic, T.G. (2008). City Logistics. In: Z.L. Chen and S. Raghavan (Eds.), *Tutorials in Operations Research 2008. State-of-the-Art Decision Making Tools in the Information-Intensive Age*, INFORMS: 181-212.

D'Arcy, L., Murray, C., & Cleave, B. (2000). The Law and Practice of International Trade. Sweet & Maxwell Ltd., 2000.

Davis, R., Shrobe, H., Szolovits, P. (1993). What is a Knowledge Representation?, *AI Magazine*, 14(1): 17-33.

DeGroot, M.H. (1974). Reaching a Consensus. Journal of the American Statistical Association, 69:118-121.

Donelly, M. (2003). Layered mereotopology. In: G. Gottlob & T. Walsh (Eds.), *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*: 1269-1274.

Donelly, M. (2004). A formal theory for reasoning about parthood, connection, and location. *Artificial Intelligence*, Vol.160/1–2: 145–172.

Eclipse (2001). Web Tools Platform Users Guide. http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.jst.ws.doc.user%2Fconcepts%2Fcwsstandards.html

Eilenberg, S., MacLane, S. (1945). General theory of natural equivalences. *Transactions of American Mathematical Society*, Vol.58/2: 231-294.

Erl, T. (2007). SOA: Principles of Service Design. Prentice Hall, 2007.

Ermolayev, V., Keberle, N., & Plaksin, S. (2004). Towards a Framework for Agent-Enabled Semantic Web Service Composition. *International Journal of Web Services Research*, Vol.1/3: 63-87.

Feigenbaum, L., Herman, I., Hongsermeyer, T., Neumann, E., & Stephens, S. (2007). The Semantic Web in Action, *Scientific American*, Vol. 297, 2007/XII: 90-97.

Fernandez-Lopez, M., Gomez-Perez, A., & Juristo, N. (1997). Methontology: From ontological art towards ontological engineering. In: *Proceedings of the AAAI Spring Symposium on Ontological Engineering*, Vol.1: 33–40.

Fernández-Rañada, M., Gurrola-Gal, F.X., & López-Tello, E. (2000). 3C - A Proven Alternative to MRP II for Optimizing Supply Chain Performance. Boca Raton, CRC Press 2000.

Fonseca, F.T., Egenhofer, M., Agouris, P., & Câmara, G. (2002). Using Ontologies for Integrated Geographic Information Systems, *Transactions in GIS*, 6(3): 231-257.

Fumagalli, L., Pala, S., Garetti, M., & Negri, E. (2014). Ontology-based modeling of manufacturing and logistics systems for a new MES architecture, *IFIP Advances in Information and Communication Technology*, 438(2014): 192-200.

Galton, A., and Mizoguchi, R. (2009). The Water Falls but the Waterfall does not Fall: New perspectives on Objects, Processes and Events. *Applied Ontology*, Vol.4/2: 71-107.

Garetti, M., & Fumagalli, L. (2012). P-PSO ontology for manufacturing systems, *Proceedings of the 14th IFAC Symposium on Information Control Problems in Manufacturing*: 247-254.

Garetti, M., & Macchi, M. (2003). The role and trends of modeling and simulation in managing manufacturing complexity, *Proceedings of 7th IFAC Workshop on Intelligent Manufacturing Systems* (2003): 1-8.

Garetti, M., Fumagalli, L., Lobov, A., & Martinez Lastra, J.L. (2013). Open automation of manufacturing systems through integration of ontology and web services. In: *Proceedings of the 7th IFAC Conference on Manufacturing Modelling, Management, and Control*: 198-203.

Gorbatov, V.A. (1979). Semantic theory of automata design (in Russian). Moscow: Izdatielstvo Energija, 1979.

Grenon, P. (2003a). The Formal Ontology of Spatio-Temporal Reality and its Formalization. *AAAI Technical Report SS-03-03*.

Grenon, P. (2003b). BFO in a Nutshell: A Bi-categorial Axiomatization of BFO and Comparison with DOLCE. *IFOMIS Report 2003/6*.

Grenon, P., & Smith., B. (2004). SNAP and SPAN: Towards Dynamic Spatial Ontology. *Spatial Cognition and Computation*, Vol.4/1: 69-103.

Grenon, P. (2006). Temporal Qualification and Change with First–Order Binary Predicates. In: B. Bennett, & C. Fellbaum (Eds.), *Proceedings of the 4th International Conference on Formal Ontology in Information Systems (FOIS'2006)*: 155-166.

Gruber, T.R. (1993). A translation approach to portable ontology specifications, *Knowledge Acquisition*, Vol. 5/2: 199-220.

Gruber, T.R. (1995). Toward principles for the design of ontologies used for knowledge sharing, *International Journal of Human-Computer Studies*, Vol. 43/5-6: 907-928.

Gutierrez, C., Hurtado, C.A., Vaisman, A. (2007). Introducing Time into RDF. *IEEE Transactions on Knowledge and Data Engineering*, Vol.19/2: 207-218.

Hahmann, T., & Grüninger, M. (2012). Region Based Theories of Space: Mereotopology and Beyond. In: S.M. Hazarika (Ed.) *Qualitative Spatio-Temporal Reasoning: Trends and Future Directions*. Hershey: IGI Global: 1-62.

Hahn, M.G., Motz, R., Pardo, A., & Musicante, M.A. (2013). Formal semantics and expressiveness of a web service composition language. In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing (SAC'13)*: 1667-1773.

Heisig, G. (2002). Nervousness in Material Requirements Planning Systems. In: G. Heisig, *Planning Stability in Material Requirements Planning Systems*, Berlin-Heidelberg: Springer, pp. 21-64.

Henzinger, T.A., Horowitz, B., & Kirsch C. M. (2001). Giotto: A time-triggered language for embedded programming. *Proceedings of EMSOFT'2001, LNCS*, Vol.2211. Tahoe City: Springer-Verlag.

Herre, H. (2010). General Formal Ontology (GFO): A Foundational Ontology for Conceptual Modelling. *Media*, Vol.2/15: 1-50.

Hitzler, P., & Janowicz, K. (2011). Semantic Web Tools and System. *Semantic Web*, Vol.2/1: 1–2.

Hoehndorf, R., Ngonga Ngomo, A., & Herre, H. (2009). Developing Consistent and Modular Software Models with Ontologies, *Proceedings of the 8th SoMeT_09 International Conference on New Trends in Software Methodologies, Tools and Techniques*, IOS Press Amsterdam: 399-412.

Höller, J., Tsiatsis, V., Mulligan, C., Karnouskos, S., Avesand, S., & Boyle, D. (2014). From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence. Academic Press, 2014.

Ingarden, R. (1964). Time and Modes of Being. Springfield: *Charles C. Thomas*.

Jacobs, F.R., Berry, W.L., & Whybark, D.C. (2010). Manufacturing Planning and Control Systems for Supply Chain Management (6th edition). IRWIN, 2010.

Jarrar, M., & Meersman, R. (2009). Ontology engineering – The DOGMA approach. In: T. Dillon, E. Chang, R. Meersman, & K. Sycara (Eds.), *Advances in web semantics I, LNCS* Vol.4891: 7–34. Berlin/Heidelberg: Springer.

Jennings, N.R., & Wooldridge, M. (1998). Applications of Intelligent Agents. In: N.R. Jennings & M.Wooldridge (Eds.), *Agent Technology: Foundations, Applications and Markets*, Springer: pp. 3-28, 1998.

Johnson, T.E. (2002).Export/Import: Procedures and Documentation. AMACOM, 2002.

Johnston III, S.L. (2015). Consequences of Insomnia, Sleepiness, and Fatigue: Health and Social Consequences of Shift Work. Medscape CME: *http://cme.medscape.com/viewarticle/513572_2*. Accessed 2015.IV.01.

Kahneman, D., & Tversky, A. (2008). Advances in Prospect Theory: An Analysis of Decision under Risk. In: D. Kahneman & A. Tversky (Eds.), *Choices, values and frames*. Cambridge University Press: 44-66.

Kuratowski, K. (1922). Sur l'opération A^{-} de l'Analysis Situs. *Fundamenta Mathematicae*, Vol.3: 182-199.

Lee, E.A. (2006). The problem with threads. *Computer*, Vol.39/5: 33–42.

Lee, E.A. (2011). Cyber Physical Systems: Design Challenges, *Proceedings of the 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*: 363-369.

Lee, H.L., Padmanabhan, V., & Whang, S. (1997). The Bullwhip Effect in Supply Chains. *Sloan Management Review*, 1997/Spring: 93-102.

Leśniewski, S. (1992). Collected Works, Dordrecht, Kluwer, 1992.

Levison, M. (2006). The Box: How the Shipping Container Made the World Smaller and the World Economy Bigger. Princeton University Press, 2006.

Lev-Ram, M. (2014). It's a subscription economy and your you are just living in it. *Fortune, 6.VI.2014. http://fortune.com/2014/06/06/welcome-to-the-subscription-economy/*

Lin, Y., & Liu, S.F. (2006). Grey information: Theory and Practical Applications, London, Springer.

Lis, S., Santarek, K., & Strzelczak, S. (1994). Organization of Flexible Manufacturing Systems (in Polish). Warsaw: WNT, 1994.

Lobov, A., Ubis Lopez, F., Villasenor Herrera, V., Puttonen, J., & Martinez Lastra, J.L. (2009). Semantic Web Services Framework for Manufacturing Industries, *Proceedings of the IEEE International Conference on Robotics and Biomimetics (2009)*: 2104-2108.

Luczak, H., Eversheim, W., Schotten, M. (2001). Produktionsplanung und –steuerung: Grundlagen, Gestaltung, Konzpete. Springer, 2001.

Macal, C.M., & North, M.J. (2006). Tutorial on Agent-Based Modelling and Simulation Part2: How to model with Agents. *Proceedings of the IEEE Winter Simulation Conference*, pp.73-83.

Mahoney, R.M. (1997). High-Mix Low-Volume Manufacturing. Upper Saddle River: Prentice Hall, 1997.

Man, K.L., & Schiffelers, R.R.H. (2006). Formal specification and analysis of hybrid systems. PhD Thesis, Technische Universiteit Eindhoven, 2006.

Maniraj, V., & Sivakumar, R. (2010). Ontology Languages – A Review. *International Journal of Computer Theory and Engineering*, 2/6: 887–891.

Marshall, C.C., & Shipman, F.E. (2003). Which Semantic Web?. Proceedings of the 14th ACM conference on Hypertext and Hypermedia HYPERTEXT'03: 57-66.

Martinez Cruz, C., Blanco, I.J., & Amparo Vila, M. (2012). Ontologies Versus Relational Databases – are they so different? A comparison. *Artificial Intelligence Review*, Vol.38: 271-290.

Martinez Lastra, J.L., Delamer, I.M., & Ubis, F. (2010). Domain Ontologies for Reasoning Machines in Factory Automation. *ISA/O3neida Automation Series, International Society of Automation*.

Matsokis, A., & Kiritsis, D. (2010). An ontology-based approach for product lifecycle management, *Computers in Industry* Vol.61/8: 787–797.

Matuszek, C., Cabral, J., Witbrock, M., & DeOliveria, J. (2006). An Introduction to the Syntax and Content of Cyc. In: *Proceedings of the 2006' AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*.: 44-49.

McKinnon, A. (2007). Road transport optimization. In: D.Waters (Ed.), *Global Logistics: New Directions in Supply Chain Management*, Kogan Page, 2007.

Mendes, M., Leitão, P., Restivo, F., & Colombo, A. (2009). Service-Oriented Agents for Collaborative Industrial Automation and Production Systems. In: *Proceedings of the Holo-MAS'2009*, LNAI 5696: 13-24.

Moles, P., & Terry, N. (1999). The Handbook of International Financial Terms. Oxford University Press, 1999.

Montreuil, B. (2011). Towards a Physical Internet: Meeting the Global Logistics Sustainability Grand challenge, *CIRRELT-2001-03 Research raport*.

Montreuil, B. (2012). Physical Internet Manifesto. *http://www.physicalinternetinitiative.org/Physical%20Internet%20Manifesto_ENG_Version%201.11.1%202012-11-28[1]*

Mörchen, F. (2006). A better tool than Allen's relations for expressing temporal knowledge in interval data. In: T. Li, C. Perng, H. Wang, and C. Domeniconi (Eds.), *Proceedings of the Workshop on Temporal Data Mining at the 12th ACM SIG KDD International Conference on Knowledge Discovery and Data Mining (KDD'06)*: 25-34.

Mormann, T. (2012). On the mereological structure of complex states of affairs. Synthese, Vol. 187/2: 403-418.

Nicholas, J.D. (1998). Competitive Manufacturing Management: Continuous Improvement, Lean Production, Customer-Focused Quality. McGraw-Hill, 1998.

Niles, I., & Pease., A. (2001). Towards a standard upper ontology. In C. Welty and B. Smith (Eds.), *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS'2001)*: 2–9.

Noy, N. F., & McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. *Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880*.

Omicini, A., & Viroli, M. (2011). Coordination models and languages: from parallel computing to self-organisation. *The Knowledge Engineering Review*, Vol.26/1: 53-59.

Österle, H., Fleisch, E., Alt R. (2001). Business Networking. Shaping Collaboration Between Enterprises. Springer, 2001.

Papadopoulos, G., & Arbab, F. (1998). Coordination models and languages. In: M. Zelkowitz (Ed.), *Advances in Computers - The Engineering of Large Systems*, Academic Press, Vol.46: 329–400.

Pěchouček, M., Rehák, M., Charvát, P., Vlček, T., Kolář M. (2007). Agent-based Approach to Mass-Oriented Production Planning: Case Study. *IEEE Transactions on Systems, Man, and Cybernetics*, Part C, Vol. 37/3: 386-395.

Peck, H. (2006). Reconciling supply chain vulnerability, risk and supply chain management. *International Journal of Logistics: Research and Applications*, 9/2: 127–142.

Pinedo, L.M. (2012). Scheduling: Theory, Algorithms, and Systems. Springer, 2012.

Proud, J.F. (2007). Master Scheduling. John Wiley & Sons, 2007.

Ramis, B., Gonzalez, L., Iarovyi, S., Lobov, A., Martinez Lastra, J.L., Vyatkin, V., & Dai, W. (2014). Knowledge-based web service integration for industrial automation, *Proceedings of the 12th IEEE International Conference on Industrial Informatics (INDIN)*: 733-739.

Resnik, P.(1999). Semantic similarity in a Taxonomy: An Information Based Measure and its Applications to Problems of Ambiguity in Natural Language. *Journal of Artificial Intelligence Research*, Vol.11/VII: 95-130.

Rosse, C., & Mejino Jr., J.L.V. (2003). A reference ontology for biomedical informatics: the Foundational Model of Anatomy, *Journal of Biomedical Informatics*, 36(6): 478-500.

Rostworowski de Diez Canseco, M., Iceland, H.B. (1999). History of the Inca Realm. Cambridge: Cambridge University Press.

Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P. (2005a). Workflow Data Patterns: Identification, Representation and Tool Support. In: L. Delcambre, C. Kop, H.C. Mayr, J. Mylopoulos, and O. Pastor (Eds.), *Proceedings of the 24th International Conference on Conceptual Modeling (ER'2005)*: 353-368.

Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P. (2005b). Workflow Resource Patterns: Identification, Representation and Tool Support. In: O. Pastor and J. Falcao e Cunha (Eds), *Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05)*: 216-232.

Russell, N., ter Hofstede, A.H.M., van der Aalst, W.M.P., Mulyar, N. (2006). Workflow Control-Flow Patterns - A Revised View. *BPM Center Report*, BPM-06-22.

Seidewitz, E. (2003). What models mean, *IEEE Transactions on Software Engineering*, Vol. 20/5: 26-32.

Sendler, U. (Ed.) (2013). Industrie 4.0 - Die Beherrschung industrieller Komplexität mit SysLM, Springer.

Shadbold, N., Hall, W., & Berners-Lee, T. (2006). The Semantic Web revisited. *IEEE Transactions on Intelligent Systems*: 96-101.

Silver, E.A., Pyke, D.F., & Peterson R. (1998). Inventory Management and Production Planning and Scheduling. John Wiley & Sons, 1998.

Slack N., Lewis M. (2011). Operations Strategy. Financial Times – Prentice Hall, 2011.

Smith, B. (1996). Mereotopology: A Theory of Parts and Boundaries, Data and Knowledge Engineering, 20: 287–303.

Smith, B. (2006). Against Idiosyncrasy in Ontology Development. In: B. Bennett and C. Fellbaum (Eds.), *Proceedings of the 4th International Conference on Formal Ontology in Information Systems (FOIS'2006)*: 15-26.

Srzednicki, J.T.J., & Stachniak, Z. (Eds.) (1998). Leśniewski's Systems: Prototothetic. Dordrecht: Kluwer Academic Publishers, 1998.

Srzednicki, J.T.J., Stachniak, Z., & Czelakowski, J. (Eds.) (1984). Leśniewski's System: Ontology and Mereology. The Hague/Wrocław: Martinus Nijhoff/Ossolineum, 1984.

Stevens, R. (2011). Closing Down the OpenWorld: Covering Axioms and Closure Axioms. *http://ontogenesis.knowledgeblog.org/1001* (12.II.2011).

Stollberg, M., & Strang, T. (2005). Integrating Agents, Ontologies, and Semantic Web Services for Collaboration on the Semantic Web. In: *Proceedings of the 1st International Symposium on Agents and the Semantic Web, AAAI Fall Symposium Series*, Arlington.

Strzelczak, S., & Berka, A. (2001). Contribution of the Theory of Parallel Computation to the Management of Distributed Manufacturing Systems. In: H. Bin, J.A. McGeough, & H. Wu (Eds.): *Computer-Aided Production Engineering*. London: PEP Ltd., pp. 29-42.

Strzelczak, S. (2012). Idiosyncratic behavior of globally distributed manufacturing. LNCS No. 398: 487-494.

Strzelczak, S. (2013). Between economics and management: chemical and biological modeling of of economic phenomena (in Polish). In: J.Brzóska and T.Pyka (Eds.), *Modernity of industries and services in the era of crisis and new challenges*, TNOiK Press, pp. 374-387.

Strzelczak, S. (2014a). Core Ontology for Manufacturing and Logistics, *Silesian University of Technology Series in Management*, 73(2014): 603-618.

Strzelczak, S. (2014b). Ontology-Aided Management, *Silesian University of Technology Series in Management*, 73(2014): 619-630.

Strzelczak, S. (2014c). Towards the **Ψ** – Production Services Internet. *IOSP Research Report*, Warsaw University of Technology, 2014.

Strzelczak, S. (2015). The TRANSFORMERS ontology for planning and controlling manufacturing and logistics operations, in systems, inter-organizational structures and operational ecosystems. *IOSP Research Report*, Warsaw University of Technology, 2015.

Sun, W., Ma, Q.Y., Gao, T.Y., & Chen, S. (2010). Knowledge-intensive support for product design with an ontology-based approach, *The International Journal of Advanced Manufacturing Technology*,48(5-8): 421–434.

Tainton, J., & Nakano, M. (2014). The Behavioral Effects of Extreme Events in Global Supply Chains. IFIP AICT, Vol. 439: 62–70.

Tarski, A. (1944). The semantical concept of truth and the foundations of semantics. *Philosophy and Phenomenological Research*, Vol.4: 341-75.

Toomey, J.W. (1996). MRP II: Planning for Manufacturing Excellence. London: Chapman & Hall, 1996.

Umeda, S. (2008). Supply Chain Simulation Combined Discrete-event Models with System-dynamic models. In: *Proceedings of the IFIP International Conference on Advanced Production Management Systems 'Innovation in Networks'* (APMS'2008): pp.563-571.

Unland, R. (2015). Industrial Agents. In: P. Leitão & S. Karnouskos (Eds.) *Industrial Agents: Emerging Applications of Software Agents in Industry*, Elsevier: 23-44.

Urbaniak, R. (2013). Leśniewski's Systems of Logic and Foundations of Mathematics. Berlin: Springer.

Uschold, M., & King, M. (1995). Towards a methodology for building ontologies. In: *Proceedings of workshop on basic ontological issues in knowledge sharing*, Vol.1: 1–15.

Vendler, Z. (1967). Linguistics and Philosophy. Cornell University Press, Ithaca, 1967.

Vogel-Heuser, B., Göhner, P., & Lüder, A. (2015). Agent-Based Control of Production Systems and Its Architectural Challenges. In: P. Leitão & S. Karnouskos (Eds.) *Industrial Agents: Emerging Applications of Software Agents in Industry*, Elsevier: 153-170.

Vollman, T.E., Berry, W.L., & Whybark, D.C. (1998). Manufacturing Planning and Control Systems. IRWIN, 1998.

Vrba, P., Radakovič, M., Obitko, M., & Mařik, V. (2011). Semantic technologies: Latest advances in agent-based manufacturing control systems. *International Journal of Production Research*, Vol.49/5: 1483–1496.

W3C (2004). OWL-S: Semantic Markup for Web Services. *http://www.w3.org/Submission/OWL-S/*

W3C (2005). Semantic Web Services Ontology (SWSO). *http://www.w3.org/Submission/SWSF-SWSO/*

Wiendahl, H.P. (1995). Load-Oriented Manufacturing Control. Springer, 1995.

Wirth, N. (1977). Toward a discipline of real-time programming. *Communications of the ACM*, Vol.20/8: 577–583.

Yoshida, Z. (2010). Nonlinear Science: The Challenge of Complex Systems. Springer, 2010.

Ontology-Based Modeling of Production and Logistics Systems

Marco Garetti, Elisa Negri, Luca Fumagalli

Politecnico di Milano, Milano, Italy

{marco.garetti, elisa.negri, luca1.fumagalli}@polimi.it

Abstract. Ontologies are useful tools for storing information on the physical, transformation and control aspects (i.e. knowledge domains) of a production system. The Manufacturing Systems Ontology (MSO) provides a framework for both process and discrete manufacturing (manufacture and assembly) and logistics. MSO is being implemented and tested within the Artemis eScop research project.

1 Introduction

Process production and manufacturing production are complex activities involving a great amount of information in all the phases of their life cycle that include product design, process design, production system design, production system operation and maintenance and lastly, product and production system dismissal and recovery. Furthermore, not only information is involved, but also a great amount of knowledge, which has different contents in the various phases:

- During product design, knowledge is mainly about materials properties, form features, design principles, strength behavior, future production costs, components features and availability, components supply network, and so on;
- During process design, knowledge is about product machining, chemical reactions, process routing, welding, joining, assembly processes and so on;
- During production systems design, knowledge has a more systemic perspective regarding the implementation of process routing with appropriate process workstations and transportation or transfer systems;
- Then, production system operation and maintenance phase involves another type of knowledge about the control rules that would allow the operation of the production system while maintaining the assets health and satisfying the productive target with the objectives of minimum cost and maximum customer satisfaction;
- Lastly, the dismissal and recovery phase is information and knowledge rich, involving information on product components and materials, knowledge on dismantling procedures and so on.

As such, ontologies can have a great importance in the field of industrial production thanks to their capacity to store information and knowledge in a structured and rich way, as it can be seen by the growing interest in research and application. Features of ontologies like object-orientation, abstract classification, encapsulation, inheritance, modularity, aggregation of objects are very useful to represent, store and manipulate the complex information and knowledge content of the industrial production field. This chapter deals with the use of ontologies in production system design and in production system operation and maintenance. Other phases of the industrial production life cycle, like product design, process design and product and production system dismissal and recovery are out of the scope of this chapter. Therefore only the following references are given for them: product design (Catalano et al., 2008; Kim et al., 2006), process design (Giovannini et al., 2012), product and production system dismissal and recovery (Kiritsis, 2011).

The chapter has the following structure. The next section presents the state of the art of ontologies in the manufacturing systems domain. The third section provides the requirements for the modelling of the manufacturing systems domain. The fourth section describes the structure of the Manufacturing Systems Ontology (MSO) proposed by the authors. MSO represents a follow-up of a previously developed production systems ontology (P-PSO) (Garetti and Fumagalli, 2012). Then, the fifth section depicts the role of MSO in the scope of the Artemis EU funded project eScop.

2 Overview of Ontologies in the Manufacturing Systems Domain

Production systems are characterized by an inner complexity and activities related to their design and management are also complex, under optimized and based largely on practices and rules of thumb. Much research and practical activities have been dedicated to the development of methods and tools to address the related problems, and a large literature of books and scientific papers have been written on these issues.

One of the main difficulties affecting the manufacturing systems domain is the extreme variety of the configurations that manufacturing systems can assume. Consequently, the efforts of many researchers toward the generalization of methods for the design and management of manufacturing systems have been often limited by the variety of the context of their applications. This is the reason why, independently from the particular design or management issue addressed, the description, classification and taxonomy of manufacturing systems are important matters for the development of effective solutions in their design and management.

Obviously, many authors have addressed the discussed issue. However, it is only with the development of information technology tools, and especially with Object Orientation (OO) concepts and related technologies (Shlaer and Mellor, 1988; Blaha and Rumbaugh, 2005; UML1.4.2 - ISO/IEC 19501, 2005) that more power has become available for better handling of these problems. In fact, this allows to abstract classifications (describing only interesting details of the objects, ignoring the minor ones), encapsulate details (hiding unnecessary details), build modularity (elements of

the model highly decoupled, but consistent), aggregate objects (compose objects from other objects).

In principle, the definition of a sound taxonomy of manufacturing systems, accordingly implemented using an ontology tool, can be the foundation of a numerous different manufacturing applications supported by information technology. In fact, if this taxonomy does not reflect a specific manufacturing system, but describes the general structure of manufacturing systems, it will have general applicability to all instances with that entity structure by definition.

The design of a manufacturing system might rely on a clear description of the structure of complex lines and shops, made of many machines, and transportation and storage systems, by using a model supported by the OO paradigm (e.g.: Bartolotta and Garetti, 1998; Garetti et al., 1997; von Cieminski et al., 2002).

Moreover, production planning and control problems might rely on a sound description of the structure of products and of related manufacturing equipment with OO (e.g.: Cavalieri et al., 2003). Among the OO models, ontologies have been widely used in the manufacturing system area. The purposes of these ontological conceptualizations are mainly as follows:

Description of manufacturing systems

Ontologies can be used to support the description of a manufacturing system, namely, supporting its representation and modelling (Bartolotta and Garetti, 1998). One important contribution to this concern is the ISO 15926 standard that supports the description of a system and could be used as a basis for the description of manufacturing systems. The standard does not attempt to define all the possible classes of production systems, but instead it provides a small set of basic engineering classes, which can be specialized by reference to a dictionary. Further details of the discussion about consistency of ISO 15926 are reported in Garetti and Fumagalli (2012).

Modelling of manufacturing systems for simulation purposes (mainly by discrete-event simulation)

For example, Battista and Giordano (2010) proposed a modelling framework that is founded on product and process data structures, process chart, data on production processes and equipment, control policies for production and inventory management, etc. A clear distinction between physical and information layer is made therein. Even if the framework is not named as an ontology, it is clearly postulated that an ontology of a manufacturing system should consider the above-mentioned elements. Ontologies can act as foundation for allowing the possibility of linking the description phase of a manufacturing system to its discrete event simulation. For example, attempts of directly connecting modelling to simulation, via a manufacturing systems ontology, are reported in Garetti (2001), Garetti and Macchi (2003), Benjamin (2006).

Scheduling of manufacturing systems

Ontologies can be used to support the scheduling of manufacturing systems. In fact, an ontology-based scheduling system can be easily instantiated to a specific factory environment without the need of an expensive and time-consuming customization. Furthermore, thanks to the common semantic description of the system's entity structure, ontologies allow to exchange or integrate information among different instances

of manufacturing systems. For example, Cavalieri et al. (2003) used an ontological description of manufacturing systems for allowing the concurrent scheduling of a network of production facilities that have a common representation thanks to the ontology.

The aim of this chapter is to present an ontology capable of a comprehensive view of all the different aspects of manufacturing systems, taking into account the requirements of the manufacturing domain modelling and the physical structure and the process view of the system. With this target in mind and based on P-PSO (Politecnico di Milano–Production Systems Ontology) (Garetti and Fumagalli, 2012), as main background, a representation of a manufacturing reality that could be used as the foundation of shop-floor control systems has been developed. The ontology uses the name of MSO (Manufacturing Systems Ontology). In particular, the version of the MSO presented in the remainder of the chapter, is dated September 2014[1].

3 Requirements for the Modelling of the Manufacturing Systems Domain

Given the complexity of production systems and considering the importance of their modelling for the manufacturing field, it is necessary to analyse the needs and requirements for this modelling activity. Research papers might suggest some requirements, such as:

- Colledani et al. (2008), who suggest that a manufacturing domain representation should be: flexible, extensible, scalable, integrated;
- Pulido et al. (2006), who state that any representation must have a compact syntax, a well-defined formal semantics, be highly intuitive, be able to represent human knowledge (in a conceptual model), have potential to build a knowledge base, have a proper link with existing web standards to ensure interoperability and to describe meaning in machine-readable way. Moreover, reasoning functionality must be considered.
- Negri et al. (2014), who indicate that the modelling must also ensure that the reasoning is possible and, moreover, it must be time-decidable and, possibly, time efficient, because in real world implementations it is not acceptable that the answer to knowledge inference does not come within a fixed time.

As for functional requirements, the modelling should be able to express all objects in the manufacturing system, their attributes and their relationships. To this end, the description of manufacturing systems should allow to define the characteristics of a class (such as a *machine*) and then to detail it in the different types (milling machine, drilling machine, and so on) without the need to re-specify the common features.

[1] The described Manufacfturing System Ontology (MSO) is the 2.0 version of September 2014, developed within the eScop research project, with special reference to the industrial use cases of the project itself. MSO is described using the Unified Modeling Language (UML). Within the eScop research project, it has been implemented using the Visual Paradigm tool.

Table 1. Requirements for the semantic language (Negri et al., 2014)

Perspectives	Language requirements
A: In order to perform automated reasoning	1. Be flexible
	2. Use formal semantics
	3. Dialog with services
	4. Include the proper links with web standards
	5. Include reasoning properties
	6. Have time-definite reasoning
	7. Have time-efficient reasoning
B: In order to ensure interoperability	1. Integrate different aspects
	2. Be machine-readable
C: In order to support the knowledge base	1. Have the potential to build the knowledge base
	2. Have persistence in the data stored
D: In order to be easy to use/maintain	1. Be extendible
	2. Be scalable
	3. Have a compact syntax
	4. Be intuitive to humans
	5. Be a consistent human knowledge representation
	6. Be object-oriented
	7. Have inheritance properties

Overall, Table 1 illustrates what are the requirements for the manufacturing domain ontology under different perspectives as documented by Negri et al. (2014).

Another important requirement regards persistency of the storage capability (information should not be lost in the case of a system crash), for this reason a secure and persistent knowledge base must support the language and used tools. Based on the work done in Negri et al. (2014), the languages that fulfil all the requirements expressed in Table 1 are the following: DAML (DARPA Agent Mark-up Language), OWL (web Ontology Language). While the OWL sub-languages are: OWL Lite, OWL DL, C-OWL (Context- OWL), OWL-Eu, OWL-E.

4 The Manufacturing Systems Ontology (MSO)

MSO is a domain ontology for the representation of manufacturing systems (both discrete and process manufacture) and logistics systems. It is based on P-PSO (Politecnico di Milano - Production Systems Ontology), which development started in the '90s as a taxonomy of manufacturing systems (Bartolotta et al., 1999; Garetti and Fumagalli, 2012). P-PSO has been based on a structured representation of the domain of manufacturing systems, supported by the object-oriented methodology, enabling the description of the relevant aspects of a generic manufacturing system. P-PSO is a *metamodel* of the manufacturing systems domain, since it specifies the entities (build-

ing blocks) it is made of, together with their main attributes, thus defining a standardized data format for describing manufacturing systems. The P-PSO only addressed the discrete manufacturing domain, also considering some logistics. In this sense, among other differences between the two models, the MSO is an evolution of P-PSO that includes also process industries and logistics.

Following the approach of P-PSO, the modelling method of MSO defines a manufacturing system from the process and logistics point of view by addressing three main different aspects separately, i.e.:

- the physical aspect, which contains the material definition of the system including workers, production facilities, material-handling and transportation equipment, storage and other supplementary devices (such as tools, jigs and fixtures);
- the technological aspect, which defines the transformational (functional) view of the system, considering the conversion processes (i.e. manufacture and assembly) and the routing that products must undergo within the manufacturing system;
- the control aspect, which in P-PSO defined the operating procedures of production at an abstract level, describing the so-called management cycle (i.e. planning, scheduling and control activities), while in MSO it should provide the data fields that are necessary for the control activity which is performed by an orchestrator engine[2].

Indeed, in MSO, the control aspect is only defined at a very high conceptual level, because differently from the other two aspects, the control structure it is not standardized enough in literature and in industrial application, thus preventing the possibility to address a generalized data structure. For this reason, only a sort of black box is determined in the overall MSO structure. It contains all the elements that are useful for the control of the production system. This black box leaves visibility only to those relationships that exit the box, while what is inside must be defined each time the control structure needs to be tailored in the specific application scope.

In addition, the visualization aspect is addressed in a separate section of the ontology, providing modelling of elements that are needed for a proper management of visualization of the manufacturing systems to enhance awareness by operators of the system control features.

5 Architecture and Classes of MSO

In this section, the class structure of MSO is described using the UML language, through the Visual Paradigm software tool (www.visual-paradigm.com)[3].

The main classes of the MSO ontology are:

- *part class*, modelling the product that has to be produced;

[2] This is the Orchestration layer in the eScop research project.

[3] Within the eScop research project, the operating implementation of MSO has been made by translating its UML description into OWL.

- *component class*, modelling the physical aspect (domain) of the manufacturing system;
- *routing operation class*, modelling the technological aspect (domain) of the manufacturing system;
- *operator class*, modelling the activity of workers / human operators; and
- *subsystem class*, a service class allowing the grouping of objects of the other classes in a nested way.

These classes are described in the following sections; for the sake of simplicity, attributes are not mentioned.

6 Modelling Products

The part class is used for modelling products. A product is seen as composed of sub-assemblies and/or items, in turn a sub-assembly, in recursively composed of sub-assemblies and/or items; this structure models the product BOM of which the product is made. A part can be either a generic part (a family of parts) or a specific part (a variant of a generic part) (Fig. 1). Therefore, part and product are synonymous and will be used consequently followingly.

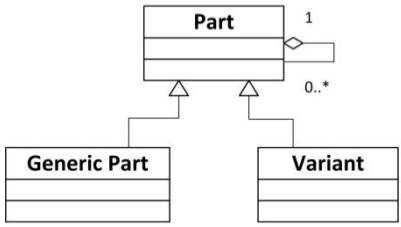


Fig. 1. Part class object model with variants

7 Modelling the Physical Aspect

The physical aspect is based on the ***Component*** class, which specializes in main and secondary sub-classes. The ***Component*** class cannot be further divided, and it is used for modelling the physical elements of a manufacturing system. The main specializations of the ***Component*** class are the sub-classes: ***Operator, Processor, Transporter*** and ***Storage***. Secondary sub-classes are ***Container, Sensor, Tool*** *and* ***Fixture***. The above-mentioned sub-classes are detailed in the following subsections.

7.1 Processor

Processor class identifies entities performing a manufacturing process function by using an energy source and/or an operator activity; for example, types of processors are: machining centers, inspection devices, assembly machines, assembly tables dedi-

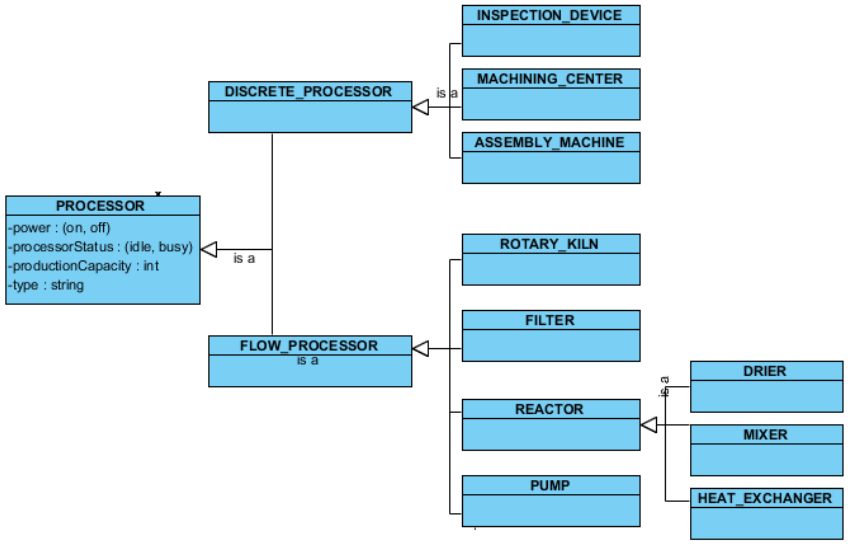


Fig. 2. Processor class and related sub-classes

Table 2. Subclasses of the Processor class

Sub-classes of the *Flow Processor* class	Sub-classes of the *Discrete Processor* class
Rotary Kiln	*Assembly Machine*
Filter	*Inspection Device*
Pump	*Machining Centre*
Reactor (can be further specialized in Heat Exchanger, Drier, Mixer)	*Assembly worktable*

cated to the function of processing a product, where processing means any activity that results in a change of the state of the part. The processor class is specialized in two sub-classes whether the processor belongs to the discrete manufacturing (***Discrete Processor*** class) or to the process production (***Flow Processor*** class) type of production process (Fig. 1). Moreover, the two classes are further specialized into specific sub-classes (Table 2).

7.2 Transporter

Transporter class identifies entities performing a transportation (i.e. logistics) function, like moving material between different points of the manufacturing process; for example: AGVs, conveyors, fork trucks and other manual or automated transport equipment for transferring and handling material, components, work pieces and assemblies to various locations throughout the factory (Table 3). In addition, the transporter class is specialized in two sub-classes (Fig. 3), whether the transporter equipment is provided for operation in the discrete manufacturing (***Discrete Transporter*** class) or in the process production area (***Flow Transporter*** class).

Table 3. Subclasses of the Transporter class

Sub-classes of the *Flow Transporter* class	Sub-classes of the *Discrete Transporter* class
Valve	*Unidirectional conveyor*
Pipe	*Sequential Buffer*
Pump	*Table module*, further specialized in *Multiplexing* (many in > 1 out), *Demultiplexing* (1 in > many out) and *Crossing* (many in > many out)
	Manual trolley
	Fork Truck
	AGV

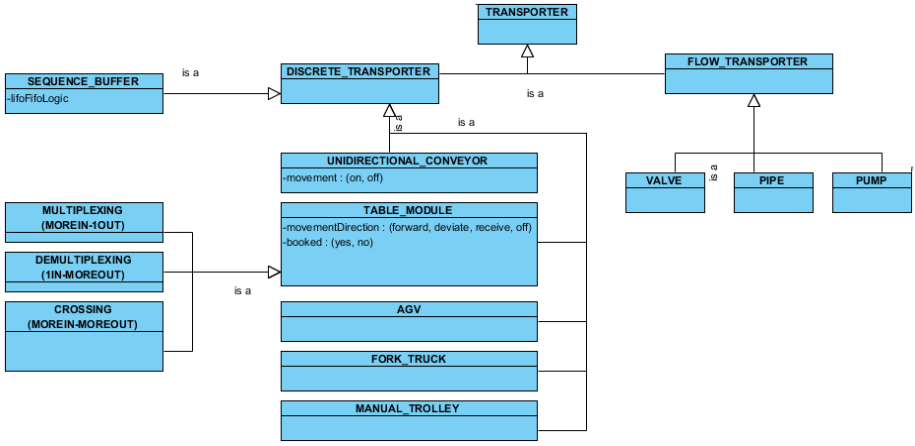


Fig. 3. Transporter class and related sub-classes

Table 4. Further subclasses of the Transporter class

Class	***Subclasses***		
Vehicle	*Guided path vehicle*	*Fixed route g. p. vehicle*	*Cart-on-track, Lift*
		Semi-fixed route g. p. vehicle	*Automatically guided vehicle (AGV), Rail mounted vehicle*
	Free path vehicle	*Non-lifting f. p. vehicle*	*Order picking cart, Pallet truck, Platform truck, Hand cart, Tractor trailer*
		Lifting f. p. vehicle	*Order picker truck, Counterbalanced truck, Stacker, Reach truck, Narrow aisle truck*
Area restricted serial transporters	*Cranes*	*Overhead traveling crane, Jib crane, Stacker crane*	
	Handlers	*Pick and place*	
		Robots	*Cartesian coordinates robot, Cylindrical coordinates robot, Polar coordinates robot, Anthropomorphous robot, S.C.A.R.A.*

A pump can be classified both under the ***Transporter*** and under the ***Processor*** classes and accordingly ***Pump*** class is linked to both ***Transporter*** and ***Processor*** class. This depends then on the function the pump performs in the particular system: in some cases, it will have a transformational role, while in others the pressure that it gives is only used for the transportation of the material. For this reason, it has a double paternity.

The MSO can be extended to include also other types of transporters, according to the specialization illustrated in Table 4.

7.3 Storage

Storage class identifies entities performing a storage function (Fig. 4), which means that they are able to keep material for later use in the manufacturing process; types of storage are, for example, tanks, buffers, automated storage and retrieval systems, dedicated to the storage (more or less temporary) of the materials in process or work-pieces. In addition, the Storage class is specialized into Flow and Discrete storage, whose further specializations are detailed in Table 5.

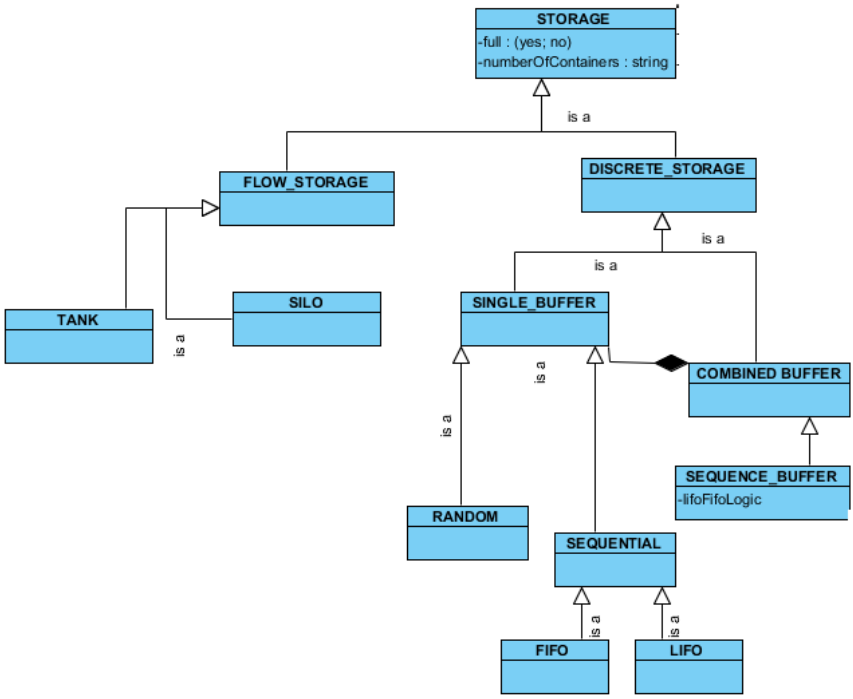


Fig. 4. Storage class and related sub-classes

Table 5. Subclasses of the Storage class

Sub-classes of the *Flow Storage* class	Sub-classes of the *Discrete Storage* class	*Sub-sub-classes*	
Tank	*Single Buffer*	*Random*	
		Sequential	*FIFO*
			LIFO
Silo	*Combined Buffer*	*Sequential Buffer*	

The Sequence Buffer is characterized by a double function: it both moves the materials or the products, but also stocks them in the waiting time between two operations. In addition, the Storage class can be expanded into more specializations by using the Combined Buffer in appropriate ways such as detailed in Table 6.

Table 6. Further subclasses of the Storage class**.**

Combined buffer	*Modeling of the combined buffer*
Block stacking	It is made up of an aggregation of l*w LIFO buffers h unit loads tall. If w>2
Stacking frame	It is made up of an aggregation of l*w LIFO buffers h unit loads tall. If w>2
Single-deep selective rack	It is made up of an aggregation of h random buffers (rows) of l cells
Double-deep rack	It is made up of an aggregation of h*l LIFO buffers of two unit load positions
Drive-in rack	It is made up of an aggregation of h*l LIFO buffers (rows) of w unit load positions
Drive-through rack	It is made up of an aggregation of h*l FIFO buffers (rows) of w positions
Pallet flow rack	It is made up of an aggregation of h*l FIFO buffers (rows) of w positions
Push-back rack	It is made up of an aggregation of h*l LIFO buffers (rows) of w positions
Mobile rack	It is made up of an aggregation of w*h random buffers (rows) of l cells
Horizontal Carousel	It is made up of an aggregation of l random buffers (columns) of h cells
Vertical Carousel	It is made up of an aggregation of h random buffers (rows) of l cells
Independent Rotating Rack	It is made up of an aggregation of l random buffers (columns) of h cells
Bin shelving	It is made up of an aggregation of h random buffers (rows) of l cells
Drawers in cabinet	It is made up of an aggregation of l random buffers (columns) of h cells (drawers)
Carton Flow Rack	It is made up of an aggregation of h*l FIFO buffers (rows) of w positions
Mobile storage	It is made up of an aggregation of w*h random buffers (rows) of l cells

7.4 Tool class

Tool class comprises the tools that are entities used by a processor to perform an operation on a product. They are a type of component with no further specialization in the current version of MSO.

7.5 Fixture class

Fixture class comprises those entities that are used to position, hold, support, locate and clamp the product in a three dimensional space. They correspond to a type of component with no further specialization in the current version of MSO.

7.6 Container class

Container class comprises the unit loads that are used for handling items (one or more items at one time); they can be made in different sizes / materials determined by the size, weight, geometry, and environmental requirements etc. of the handled item. The most common types used in manufacturing systems are disposable or reusable pallets, bins, boxes, baskets, etc.

7.7 Sensor class

Sensor class identifies sensing devices used for capture the status of a physical variable (Fig. 5). Sensors can be of different types: ***Photocell, Barcode Scanner, Temperature Sensor, Conductivity Sensor, Moisture Sensor, Tank Level Sensor, Pressure Sensor, Flow Meter***, etc. that correspond to the sub-classes of the ***Sensor*** class.

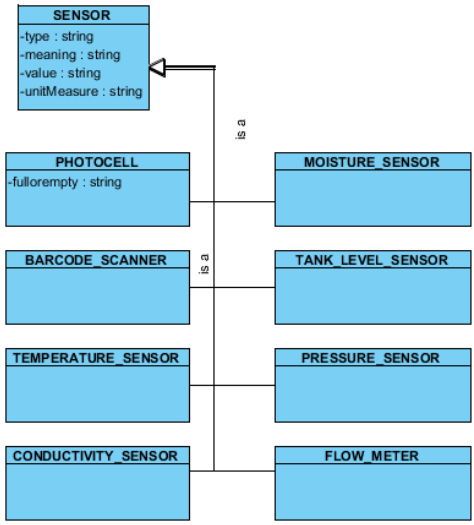


Fig. 5. Sensor class and related sub-classes

7.8 Operator class

Operator class identifies workers that perform activities in the manufacturing system and interact with other components (Fig. 6). This class needs a specific approach, since operators can carry out different types of operating activity (for instance processing, transport, inspection, maintenance) and carry out no-less-important control activities, like support and supervision over any type of component. The MSO ontology specializes the ***Operator*** class in ***Process Operator*** (further specialized in transportation, manufacture and assembly operator) and ***Control Operator*** (a supervisor, manager, quality control operator). The ***Operator*** class is a sub-class of the component class.

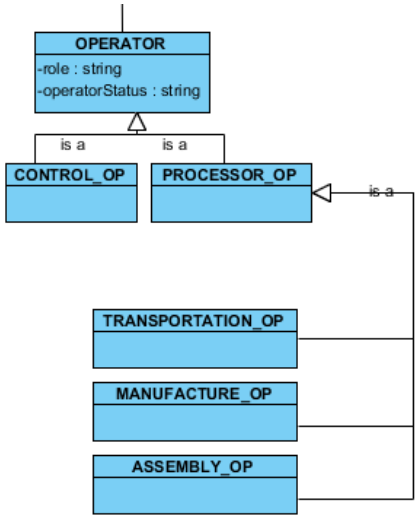


Fig. 6. Operator class and related sub-classes

8 Modelling the technological aspect

The technological aspect of manufacturing and logistics systems is based on the transformational (functional) perspective of the conversion processes that items to be manufactured undergo, following a *process plan* within the manufacturing system itself.

Before illustrating the technological aspect in detail, some assumptions must be made:

- First, *workstation* is defined as the environment in which product transformations are made possible (e.g. assembly, cutting, drilling, picking, etc.). A workstation is traditionally made of a floor space and some components, like a machine and an operator. In the MSO modelling approach, a workstation is a sub-system composed of components like a processor and an operator.

- *Operation* is defined as the technological process activity that is performed in a workstation, thus changing the product status (e.g. assembly operation, picking operation, chemical reaction, etc.)
- *Process plan* (which is also called *routing*) is defined as an ordered sequence of individual operations, which has been decided by the process engineer, for producing a product by means of the visits to a certain number of workstations. A process plan is typically represented by an operations process chart (i.e. a routing, which describe the relationships among operations through a graph) and by an operations process table (which reports technological data on the operations, like for example the type of machine required, cutting speed and so on). In manufacturing, process plans are prepared off-line by the production-engineering department by deciding the best technological solution for product manufacture taking into account the process equipment available and the tools and fixtures required. As a final output of the process planning activity, the routing is defined, together with the tooling and CNC control programs of the related machines. In some cases, routing alternatives can be included in process plans by providing optional routes at some point of the process plan so to have more flexibility.

We can distinguish two types of routing:

- Transportation routing: is made of a linear sequence of visits to workstations in each of which a manufacturing or assembly operation is performed (the allocation of operations to workstations is decided by production engineering in the prior planning stage);
- Process routing: is a more detailed version of the transportation routing in which each operation is detailed in the more elementary activities of manufacture or assembly, which are performed into each workstation.

Transportation routing is the foundation for driving the evolution of the product transformation process within the production system by means of the description of the operations that the product has to undergo for being processed (i.e. the sequence of workstations that the product has to "visit" to do so).

On the contrary, process routing contains details that are directly managed by the workstations; hence, they are not essential from the routing point of view. Therefore, they are not modeled in the MSO ontological model.

For example, in Figure 7, both transportation and process routings are represented for product X. Let suppose that product X requires to visit workstation WS1 and then WS2 to be completed. In workstation WS1 operation A is performed and in workstation WS2 operation B.

Finally, operation A is composed of activities A1 and A2, while operation B is composed of activities B1 and B2. Transportation routing is expressed by the sequence of workstation WS1 and then WS2.

Followingly, the ***Routing*** class (Fig. 8) defines the concept of transportation routing accordingly.

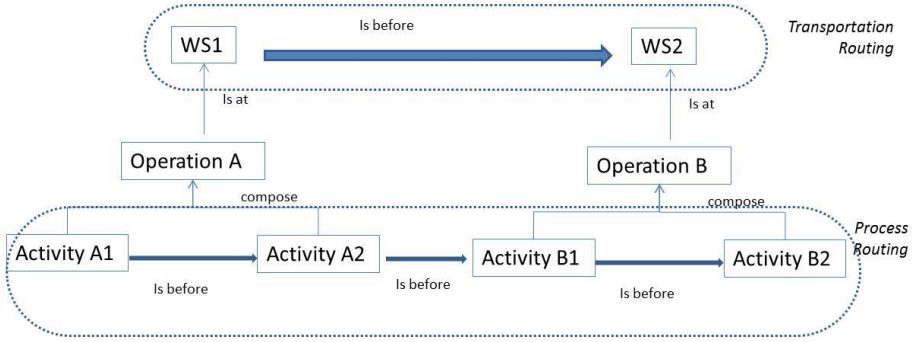


Fig. 7. Concept of transportation and process routing

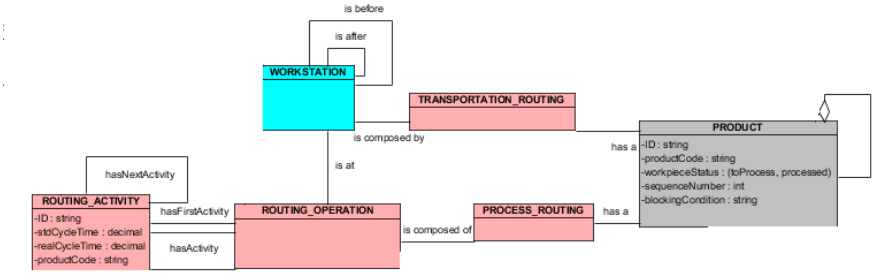


Fig. 8. UML model of transportation routing

The object-oriented view of the transportation routing is made of Workstations, each of which is characterized by a ***Routing Operation*** that can be defined as a manufacture or assembly activity (or task) which occurs at a processor and consists of a set of actions, which lead to a well-defined state change - physical or otherwise - in the product.

The same formal description of the transportation routing may be used, for both manufacture and assembly, in two different situations from an industrial engineering point of view:

- Product to workstation processing (with equipment already assigned to workstation)
- Equipment to workstation processing (with product already assigned to workstation)

Product to workstation processing means that the product has to travel among workstations to be processed and finds all necessary equipment ready to do so in the workstations (e.g. tools or fixtures for manufacture and components for assembly).

Equipment to workstation processing means that the equipment (e.g. tools or fixtures for manufacture and components for assembly) has to travel among workstations so to allow products that are already assigned to workstations, to be processed.

9 Modelling the control aspect

In P-PSO, the control aspect was defined as a set of elements like controller, rule, order, production plan, batch and task. In MSO, such classes are not present anymore in this shape. In fact, the MSO's main function is to be the knowledge base that supports the control software and, for this reason, it is not meaningful to have the controller class represented. On the other hand, it should have a set of elements that store the right information for the control software. Since there is no standardization in the control software and in what information it requires, it is not possible to build a general control aspect that would apply to any situation. In MSO, this issue is solved with the creation of a control black box in the ontology that will be defined and adapted to each implementation situation and whose only visible parts are the relationships with the elements of the physical and technological aspects.

10 Grouping objects through the subsystems class

The subsystem class denotes an aggregation of resources, which in turn can be other subsystems or elementary resources that can be objects of the component, operation and controller classes. An example of subsystem is a workstation. The subsystem class can be used for managing different levels of detail of a manufacturing system, denoting a generic group of physical resources. Moreover, the subsystem class allows users to save some parts of a manufacturing system as a black box of resources and to re-use them in a different model. The subsystem class has a fundamental role in MSO modelling (as it was also in P-PSO) since it allows creating customized building blocks that can be used everywhere in the object model. For example, Figure 9 shows the relationship between the subsystem and the component classes.

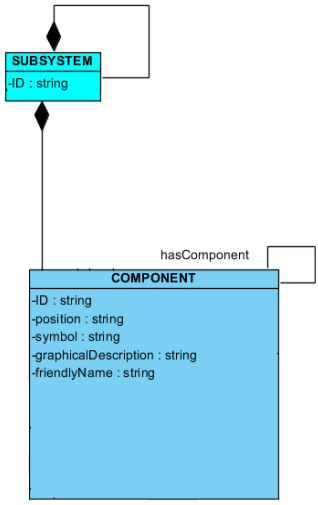


Fig. 9. Relationship between subsystem and component classes

To this concern, all components of a manufacturing system must be connected with each other (i.e. with at least one other component); the link between two components is done by instancing the association relationship between them (Fig. 9). The association linking two components must specify direction of material flow. In fact, the flow can be bi-directional or one-directional according to the possibility of flow. A consistency check can be performed in order to verify that each subsystem / component (apart from the input / output components), has at least two mono-directional or one bi-directional link. Another consistency check can be done for verifying that every processor and every storage device is linked to at least one transporter.

11 Role of MSO in the eScop architecture

MSO is one of the main components of the eScop project. The project aims at overcoming the current problems of system integration at shop-floor control level by semantically integrating embedded devices and applications based on open standards. The central concept of **eScop** is to combine the power of embedded systems with an ontology-driven service-based architecture for realizing a fully open automated manufacturing environment as depicted by Garetti et al. (2013). Indeed, the proposed solution is based on the merge of ontology knowledge, represented by MSO, with the SOA control architecture allowing the control to be automatically configured by the MSO content, while embedded systems allow this architecture to operate the shop floor equipment. Figure 10 illustrates the software architecture of the eScop control system for open automation manufacturing, in which the role of MSO can be identified. In the figure, the three main control layers of the architecture are visible: PHL denotes the PHysical Layer, RPL stands for the RePresentation Layer and ORL denotes the ORchestration Layer. In addition, the complementary INTerface (INT) and VISualization (VIS) layers complete the architecture. The layers are interconnected with round arrowhead connectors, showing information flow among the layers via web-service. The content and purpose of the layers is the following:

- The PHL (physical layer) contains service-enabled embedded devices (RTU, Remote Terminal Units) that can control the shop floor equipment;
- The RPL (representation layer) is made of 2 components: i) Ontology Service, which makes it possible to query and update the MSO knowledge base, and ii) the MSO knowledge base which is the eScop Manufacturing Systems Ontology. The eScop MSO contains all the information and relationships related to the controlled manufacturing system and it reflects its current status in real time; MSO is updated and queried for decision making by the orchestration and physical layer;
- The ORL (orchestration layer) is also made of 2 components: i) Service Composer, which receives task needs and decides how to orchestrate the system in order to fulfill task needs, and ii) Orchestrator service which does the orchestration process to ensure its successful execution;
- In addition, the visualization layer also connects to the MSO knowledge base through the ontology service to get information on visualization.

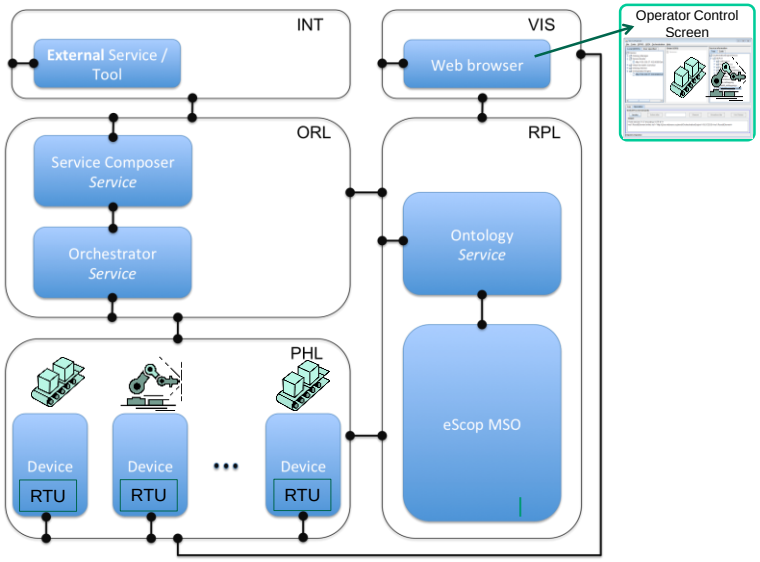


Fig. 10. Positioning of MSO in the eScop general architecture

The result is a modular, fully open software solution for the operational control of manufacturing equipment allowing the following application goals of remarkable industrial relevance:

- easy and fast commissioning of new plants,
- achievement of "plug & produce" inclusion of new equipment,
- replacement of traditional control based on hierarchical hardware architecture, by a single level cohort of embedded systems and a series of software control levels.

References

Bartolotta, A., & Garetti, M. (1998). Object-oriented representation of manufacturing systems: State of the art and perspectives. *Proceedings of the APMS'96*: 195–205.

Bartolotta, A., Corradi, E., & Garetti M. (1999). Developing an ontology for the modelling of manufacturing systems. *Proceedings of the International Enterprise Modelling Conference, IEMC'99*, Verdal, Norway.

Battista, C., Giordano, F., Iannone, R., & Schiraldi, M. (2010). A proposal for a standard framework for simulating and modelling manufacturing systems. In: S. Digiesi, G. Mossa, G. Mummolo, L. Ranieri, *Sustainable Development: Industrial Practice, Education & Research*, Vol. 2.

Benjamin, P., Patki, M., & Mayer, R. (2006). Using ontologies for simulation modeling. *Proceedings of the 38th Winter Conference Conference*: 1151-1159).

Blaha, M.R., & Rumbaugh J.R. (2005). Object – Oriented Modeling And Design With UML, Pearson - Prentice Hall.

Catalano, C.E., Camossi, E., Ferrandes, R., Cheutet, V., & Sevilmis, N. (2008). A product design ontology for enhancing shape processing in design workflows. *Journal of Intelligent Manufacturing*, Vol.20: 553–567.

Cavalieri S., Garetti M., & Terzi S. (2003). Using UML for describing a reference framework for benchmarking production scheduling systems. Madeira, Portugal. *Proceedings of the 10th International Concurrent Engineering Conference (ISPE'2003)*, 26–30 July 2003, Madeira, Portugal.

Cavalieri, S., Garetti, M., Macchi, M. (2000). Virtual Prototyping of Manufacturing Systems. Sydney. *Proceedings of the Eighth International Conference on Manufacturing Engineering (ICME'2000)*.

Colledani, M., Terkaj, W., Tolio, T., & Tomasella, M. (2008). Development of a conceptual reference framework to manage manufacturing knowledge related to prod ucts, processes and production systems. In: A. Bernard, & S. Tichkiewitch (Eds.), *Methods and tools for effective knowledge life-cycle-management*: 259-284. Springer Verlag.

Garetti, M. (2001). State of the Art and Future Perspectives of Modelling and Simulation in Manufacturing. *Proceedings of the MITIP'2001 Conference*. Plzeň, Czech Republic.

Garetti, M., & Macchi, M. (2003). The role and trends of modelling and simulation in managing manufacturing complexity. *Proceedings of the 7th IFAC Workshop on Intelligent Manufacturing Systems*. Budapest, Hungary, 6–8 April 2003.

Garetti, M., Bartolotta, A., Corradi, E., Rabe, M., & Raimondo, A. (1997). Design issues of an integrated software workbench supporting the manufacturing systems design process. *Proceedings of CAPE'97*: 320–329, Detroit, USA.

Garetti, M., Fumagalli, L. (2012). P-PSO ontology for manufacturing systems. *Proceedings of the 14th IFAC Symposium on Information Control Problems in Manufacturing (INCOM'2012)*: 1- 8, May 23-25, 2012, Bucharest, Romania.

Garetti M., Fumagalli L., Lobov A., Martinez Lastra J. L. (2013). Open automation of manufacturing systems through integration of ontology and web services. *Proceedings of 7th IFAC Conference on Manufacturing Modelling, Management, and Control*, Saint Petersburg, Russia, June 19-21, 2013.

Giovannini, A., Aubry, A., Panetto, H., Dassisti, M., & El Haouzi, H. (2012). Ontology-based system for supporting manufacturing sustainability. *Annual Reviews in Control*, Vol. 36(2): 309-317.

Kim, K.-Y., Manley, D.G., & Yang, H. (2006). Ontology-based assembly design and information sharing for collaborative product development. *Computer Design*, Vol.38: 1233–1250.

Kiritsis, D. (2011). Closed-loop PLM for intelligent products in the era of the Internet of Things. *Computer Design*, Vol. 43: 479–501.

ISO/IEC 19501 (2005). Information technology - Open Distributed Processing - Unified Modeling Language (UML) Version 1.4.2. http://www.omg.org/spec/UML/ISO/19501/PDF/. Accessed 2012.III.10.

Lin, L.F., Zhang, W.Y., & Lou, Y.C. (2009). Developing manufacturing ontologies for knowledge reuse in distributed manufacturing environment. *International Journal of Production Research*. Vol. 49/2: 343–359.

Morten, L., Roulet-Dubonnet, O. (2011). Holonic shop-floor application for handling, feeding and transportation of workpieces. *Int'l Journal of Production Re search*, Vol. 49/5.

Negri, E., Fumagalli L., Garetti, M., Tanca, L. (2014). A review of semantic languages for the conceptual modelling of the manufacturing domain. *Materials of XIX Summer School "F. Turco" A Challenge for the future: the role of industrial engineering in a global sustainable economy*. Senigallia, Italy, September 9-12, 2014.

Pulido, J.R.G., Ruiz, M.A.G., Herrera, R., Cabello, E., Legrand, S., & Elliman, D. (2006). Ontology languages for the semantic web: A never completely updated review. *Knowledge-Based Systems*, 19: 489-497.

Schlenoff, C. I., Ivester, R. W., Libes, D., Denno, P. O., Szykman, S. (1999). An Analysis of Existing Ontological Systems for Applications in Manufacturing and Healthcare. *NIST Interagency/Internal Report (NISTIR) – 6301*.

Shlaer, S., & Mellor, S.J. (1988). Object oriented systems analysis: Modelling the world in data. Prentice Hall.

Smith, B. (2006). Against Idiosyncrasy in Ontology Development, In B. Bennett and C. Fellbaum (Eds.), *Formal Ontology and Information Systems*, Baltimore.

von Cieminski, G., Wiendhal, H.P., Garetti, M., Macchi, M. (2002). Proposal of a Reference Framework for Manufacturing Systems Engineering, *Proceedings of the International Conference on Enterprise Integration and Modeling Technology* (ICEIMT'02). April 24-26, 2002, Valencia, Spain.

Classification of Knowledge Representation Implementations in the Manufacturing Systems Domain

Borja Ramis Ferrer

Tampere University of Technology, Tampere, Finland

borja.ramisferrer@tut.fi

Abstract. Ontologies are presented as a powerful mechanism for integration of components that are located in different levels of the ISA-95 automation pyramid, which is widely known in the industrial automation domain. Hence, the development of systems that use knowledge representation is a feasible manner for the reduction of efforts, e.g. in vertical communication implementation. This kind of research is challenging because of the quantity of cross-layer information exchange. In fact, as industrial automation systems are, by nature, dynamic, process control components must be capable of adapting fast to changes. Furthermore, reconfiguration of scalable systems can be automated through ontology modeling. This chapter presents an investigation on how representation of knowledge is utilized in different industrial automation developments. In addition, main concepts and requirements for designing knowledge representation implementations are identified and described. Finally, according to this description, a classification of distinct implementations is also presented.

1 Introduction and overview of the related research and work

This chapter reviews recent developments of Knowledge Representation (KR) in the industrial automation domain. The focus of this chapter is on how ontologies are being used in domain specific activities, e.g. in network configuration support and Knowledge Driven (KD) approaches, which are considered within the industrial automation domain. Moreover, this chapter discusses main requirements for using KR in the presented cases.

Design and implementation and of industrial automation systems tend to follow a hierarchical structure. Meanwhile, higher entities are in charge of planning, scheduling, supervision or monitoring tasks, while lower level subsystems play the role of executing physical actions, e.g. for assembling a product. Due to the effort that is required for integrating heterogeneous information of manufacturing systems, the ISA-95 standard appeared to enable interfacing the enterprise and control systems in industries (ISA-95).

The problem of these large-scale implementations is the integration of different layers for exchanging information in any direction of their hierarchical structure. One solution is the use of gateways or entry-points, as elements that allow both vertical

and horizontal communications. Hence, gateways give to subsystems the capability of communicating with others located in different or same levels.

Once the implemented entry-points permit the flow of information through the entire system, one efficient manner of facing the incompatibility between different subsystems that use heterogeneous information, is the utilization of the ISA-95.The described models and terminology in this international standard facilitates the communications by standardizing interfaces, formats and semantics. Then, the need of modeling systems with specific terminology becomes important in industrial automation domain implementations.

On the other hand, Artificial Intelligence (AI) (Russell & Norvig, 2010) is a field that studies the intelligent behavior of machines or software. In other words, AI is the field in which scientists and engineers create intelligent machines or intelligent computer programs.

Knowledge Representation (KR) is a part of AI concerned on allowing systems to use its knowledge and make decisions (Brachman & Levesque, 2004). In fact, a Knowledge Base (KB) is where the represented knowledge is stored. Recent research works implement knowledge-based systems in the factory automation domain. These approaches utilize a KB for storing needed information and supporting, i.e. decision-making and execution, of processes (Lastra et al., 2010; Puttonen et al., 2013a; Ramis Ferrer et al., 2014). More precisely, it can be stated that a knowledge-based system consists in two components: a KB and an inference engine. Meanwhile the KB represents facts about the world in which the system inhabit; the inference engine allows the reasoning on represented facts and creation of new ones through logic statements and/or rules. For instance, Puttonen et al. (2013b) describe an implementation in which a KB is used for describing a composite process that is later executed. Then, within AI implementation, current machines are capable to determine, by themselves, the performance of actions through the KR and implementation of a KB.

During 1980s and beginning of 1990s, manufacturing system engineers started developing expert systems using First-order Logic (FOL), which permits the world representation trough statements. It can be stated that an expert system is a knowledge-based system. Thus, it includes a domain KB and an inference engine, for supporting the decision-making. However, the configuration and design of expert systems using FOL causes high time and effort cost when expressing knowledge and integrating it with different subsystems.

Then, the reduction of effort and time demand was achieved by the creation of logic languages, which allow representing easily domain knowledge and support the KB design. Then, ontologies appeared in early 1990s. In the industrial automation domain, ontologies have been widely researched and tested for exploiting their full potential from then until now.

As it can be seen in recent research works (Garetti, 2012; Puttonen et al., 2013a; Puttonen et al., 2014; Uddin et al., 2011), the community is moving towards the use of ontologies as the mechanism for describing the knowledge of manufacturing systems. Gruber (1993) defined ontology as *"an explicit specification of a conceptualization"* and it is probably the most used definition for explaining this knowledge representation paradigm.

Principally, ontologies serve for representing any domain knowledge formally but they can also be classified depending on i.e. its structure, subject of conceptualization or task dependency (Asuncion Gomez-Perez, 2004; Lastra et al., 2010). Ontologies can be easily designed by following methodologies as, for instance, an eight-step guide available in (Noy and McGuinness, 2001). Moreover, this methodology is also described in Lastra et al. (2010).

There exist many languages for description of ontologies (Maniraj and Sivakumar, 2010), like the Web Ontology Language (OWL) (OWL, 2004), which is extensively utilized by many ontology designers. Actually OWL is a vocabulary extension of the Resource Description Framework (RDF) (RDF, 2015). Moreover, OWL is presented by Lastra & Delamer (2006) as a mature language for implementation of KBs for various systems.

On the other hand, query languages, as the SPARQL Protocol and RDF Query Language (SPARQL) (SPARQL, 2008), are used for retrieving the information from ontology models. In addition, KBs of systems can be updated, i.e. using SPARQL Update (SPARQL 1.1 Update, 2013), which enables manipulation of the RDF graph.

Examples on how SPAQRL and SPARQL Update can be utilized on industrial systems can be found in (Puttonen et al., 2013a; Puttonen et al., 2013b; Ramis et al., 2014). In fact, these research works demonstrate that knowledge-driven implementations are powerful developments that permit the control of system's KBs and factory automation processes execution on runtime.

The remaining part of the chapter is structured as follows. Second section defines two main groups for classifying the use of KR in manufacturing systems. Then third section presents a set of requirements that must be considered when designing a knowledge-based system. Finally, fourth section concludes the chapter.

2 Use of KR in recent Manufacturing Systems research

In this section a general classification of different types of KR use in manufacturing systems is presented. The identification of distinct KR implementation (KRi) types has been possible through a review of recent domain approaches. The wide variety of tasks (or activities) in which the representation of knowledge is involved, demonstrates that researchers have accepted the use of KR as an evolution for the industrial automation domain. It should be noted that this classification is technology independent, so that its objective is to provide a possible classification of KRi in the manufacturing systems domain. On the other hand, this section presents some research works as examples of each KRi type.

The KR use in manufacturing systems can be classified in three main groups (or types): Domain Specific Task implementation (DSTi), Combination of Task implementation (CoTi) and Full-KD implementations (Full-KDi). This structurization is shown in Fig.1.

In this chapter and along presented research *task implementation* is understood as a development that is needed for supporting any stage of a given manufacturing system activity.

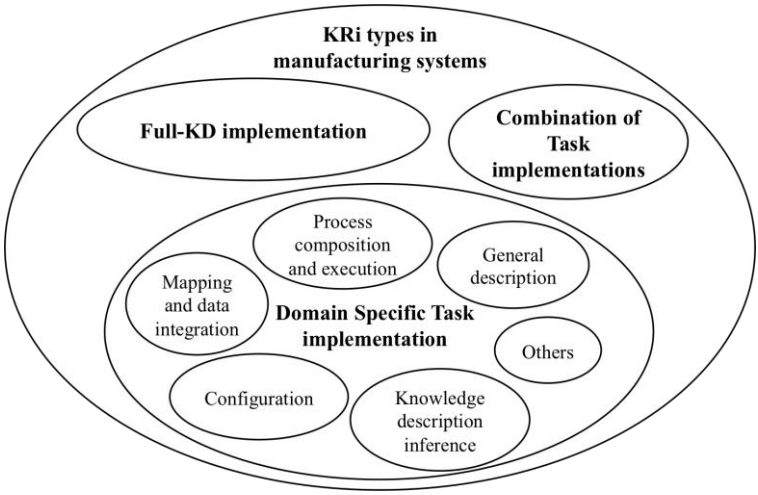


Fig. 1. Types of KR implementation (KRi) in manufacturing systems

A DSTi is understood as a task, which is performed using data represented in a KB. As it can be seen in the classification diagram, each activity belonging to the DSTi group is a task that must be performed in manufacturing systems. These activities are performed in different parts of the system, hence allowing the cross-layer exchange of information. It should be noted that the DSTi class within the diagram shown in Fig.1 could be extended by other DSTi types that are not considered in this chapter.

The ***General description*** concept includes any development that uses a KB for describing knowledge needed at some point in the manufacturing process. For example, (Garetti, 2012) describes a research work consisting in a description of manufacturing systems, proposing a general taxonomy.

The ***Knowledge description inference*** concept considers KR developments that allow any kind of support in manufacturing systems by reasoning in a KB. For instance, (Giovannini et al., 2012) describes that the developed knowledge based system uses KB inferences for stating associations that support product solutions, following defined requirements. In fact, a *knowledge description inference* activity is a *general description task*, which uses an inference engine. As another example, the research work presented in (Zhang and Luo, 2014) proposed a new ontological model, aiming at construction of an open knowledge system for engineering materials. This work defines a rule base using the Semantic Web Rule Language (SWRL).

The ***Configuration*** concept includes those developments that use KR for supporting configuration of subsystems at any level. For instance Ramis Ferrer (2013) presented a research in which the KR facilitates automatic configuration of Function Block Network, which is configured by querying ontologies used also for calculation of key performance indicators. This work was part of the PLANTCockpit project[1], which developed a platform for monitoring and controlling production processes.

[1] http://www.plantcockpit.eu/

The ***Mapping and data integration*** concept includes those developments that use KR for allowing the mapping of different domain models, which is needed for heterogeneous data integration. In fact, Ramis Ferrer also presented a development that permits the data integration by mapping different ontology domains using the ISA-95 standard terminology.

The ***Process composition and execution*** concept considers KR is that allow the generation of a process to be later executed. In an example Puttonen et al. (2013b) presented a semantic web service composition pattern based on OWL-S, which is used for service description. In addition, SPARQL and SPARQL Update are used for monitoring and controlling when the process is being executed.

It should be noted that Fig.1 presents the most relevant DSTi that have been found by researching recent KR developments in manufacturing systems. Nevertheless, it is probable that on-going small KR developments are still not disseminated. Thus the ***Others*** concept represents those DSTi that can be used in any non-defined activity using KR.

A ***CoTi*** is understood as a KRi type that includes developments that combines more than one DSTi. In fact, the research work presented in (Ramis Ferrer, 2013) is an example of a CoTi. As it has been explained before, it combines the *Configuration* and *Mapping and data integration* concepts, which are presented in Fig.1.

Finally, a ***Full-KDi*** is defined as a development that involves KR for any task on manufacturing systems. More precisely, Full-KDi is a higher-level implementation that integrates different subsystems of a manufacturing system with the use of represented knowledge. As an example of this type of KRi, the eScop project can be given[2], which intends to overcome the actual drawbacks for the shop floor control level. In (Ramis Ferrer et al., 2014) an early research work realized within the eScop project demonstrates an implementation of knowledge-based web service integration for industrial automation systems. Thus, Ramis Ferrer et al. (2014) present an example of a Full-KDi since it presents an approach that is driven by knowledge at all levels of the approach.

3 Main requirements for each type of KR use

The previous section can be used as a quick reference, including examples of recent DSTi for classification of distinct KRi in manufacturing systems. However, when facing a new development it is not enough being able to overview existing works because any KRi requires the adaptation of the KB design to the implementation objective, functionality and the integration of it with other parts of the system, which need to consume system knowledge.

This chapter describes a set of requirements that developers must take into consideration when approaching any type of KRi in the manufacturing systems domain. These requirements are: *Taxonomy definition*, *KB design*, *Query definition*, *Rule base design*, *Reasoner selection*, *Multiple models* and *Ad-hoc solution*.

[2] http://www.escop-project.eu/

Taxonomy definition is a requirement that is not only concerned about the definition of terminology that will be used in the models, but also with the hierarchy of concepts and their relation in i.e. ontological models. This is an important requirement because it determines how the data is translated to languages that are manipulated by subsystems of the manufacturing system. It should be noted that Asuncion Gomez-Perez (2004) described some recommendations to build taxonomies.

KB design is the requirement that is focused in the mechanism for storing knowledge, which must be accessible by interested subsystems. Fundamentally, it consists on defining the statements that represents the system domain, expressed using the defined terminology in previous requirement.

It should be noted that, in case of ontologies, this requirement imposes the definition of: concepts (or types) organized by a taxonomy, properties or relations between types, functions for representing operations in concepts, true statements or axioms and instances, which are real world elements belonging to a certain type. As stated in first section, Noy et al. (2001) presented an eight-step methodology for ontology design.

Query definition is needed for retrieving information and even modifying ontological models. This interaction allows knowledge-based systems i.e. monitoring actual status of the manufacturing systems and support the process control. Hence, the query definition becomes an important requirement for any KRi that needs the KB control. For instance, any process of a Full-KDi depends on the represented data so the query definition is critical for its success.

Rule base design is a requirement concerned on definition of rules for the KB of a system. Any knowledge model includes a TBox (terminological knowledge) and an ABox (assertions on knowledge). Additionally, designers can define a set of rules that are included to the model. Although rule definition is an optional definition on ontological models, it allows to inference on described statements.

Reasoner selection is needed when developers desire to use inference on ontological models. Reasoning engines are capable of inferring undefined data sets as new facts by analyzing described statements and defined rules. On the other hand, reasoning engines permit to validate models by checking its consistency. It should be noted that there are a wide variety of reasoners: Pellet, FACT++, RACER, Hermit and Snorocket, among others. The reasoner must be chosen in order to fulfill the KRi needs for managing inferences. With the objective of supporting the selection, (Dentler et al., 2011) and (Abburu, 2012) can be referred with regard to recent comparisons of available reasoners.

Multiple models requirement is needed when the system designer decides to store the required data in more than one model. For instance, it is possible that the system uses modular ontologies for representing the entire system. This means that several domain ontologies are designed for representing the system knowledge, instead of implementing a unique centralized model. Therefore, all data sets, that compose the entire model, are integrated in some point of the implementation or, simply, queried in a decentralized manner. Developers can review research works such as (Stuckenschmidt & Klein, 2003), which presents the benefits of using modularized ontologies.

Table 1. Need for requirements for developing different KR implementation types (KRi)

KRi type \ Requirements		Taxonomy definition	KB design	Query definition	Rule base design	Reasoner selection	Multiple models	Ad-hoc solution
DSTi	General description	✔	✔	✔	NA	NA	Op	Op
	Knowledge description inference	✔	✔	✔	Op	Op	Op	Op
	Configuration	✔	✔	✔	Op	Op	Op	✔
	Mapping and data integration	✔	✔	✔	Op	Op	Op	✔
	Process composition and execution	✔	✔	✔	Op	Op	Op	✔
CoTi		✔	✔	✔	Op	Op	Op	✔
Full-KDi		✔	✔	✔	Op	Op	Op	✔

✔ = Yes ; Op = Optional ; NA = Not Applicable

Ad-hoc solution is the requirement that is concerned with developing applications for solving any specific issue. There are many examples of ad-hoc solutions that can be driven by the use of manufacturing system KR as i.e.: service implementations for interaction with an ontological model, gateway development for heterogeneous subsystems integration, or runtime process monitoring and data interpretation.

Nevertheless, it should be noted that not all the KRi types have the same requirements due to their specific objectives. To clarify the KRi type needs Fig.1 presents the need of described requirements for developing different KRi types. The benefit illustrated by Table 1 is that to identify the requirements it must be fulfilled by each KRi.

As it can be seen in Table 1, meanwhile the *Taxonomy definition*, *KB design* and *Query definition* are required in any implementation using KR; the rest of requirements are needed or not depending on the KRi type.

The definition of taxonomies is needed because KB designers must represent the systems using domain specific terminology. Moreover, the requirement of designing a KB is a must for storing the system knowledge. Finally, the definition of queries also is required because any KRi require a mechanism of interacting with the KB. In this way, knowledge-based systems are capable of retrieving data of the domain model.

On the other hand, the *Rule base design* is optional for any KRi less than the *general description* because, by definition, the objective of a general description does not require the use of reasoning. Thus, the same explanation is applicable for *Reasoner selection* requirement. For the rest of KRis, the requirements that are concerned with the use of inferences are optional since it is the choice of designers the need of inferenced knowledge.

Table 2. Requirements met along recent research works on manufacturing systems

KRi type \ Requirements		Taxonomy definition	KB design	Query definition	Rule base design	Reasoner selection	Multiple models	Ad-hoc solution	Example
DSTi	General description	✔	✔	✔	✖	✖	✖	✖	(Garetti, 2012)
	Knowledge description inference	✔	✔	✔	✔	✔	✔	✔	(Dentler et al., 2011)
	Configuration	✔	✔	✔	✖	✖	✖	✔	(Ramis Ferrer, 2013)
	Mapping and data integration	✔	✔	✔	✖	✖	✔	✔	(Ramis Ferrer, 2013)
	Process composition and execution	✔	✔	✔	✔	✔	✖	✔	(Puttonen et al., 2013b)
CoTi		✔	✔	✔	✖	✖	✔	✔	(Ramis Ferrer, 2013)
Full-KDi		✔	✔	✔	✖	✖	✖	✔	(Ramis Ferrer et al., 2014)

✔ = Yes ; ✖ = No

Because ontologies allow modularity, it has been stated that some developments can implement KR within decentralized approaches. Thus, the *Multiple models* development will be a requirement of the KRi if the knowledge is not located in a unique domain model. Moreover, the need of *Ad-hoc solutions* is required in almost all the KRi. The reason is that engineers must develop domain specific solutions for interacting with the ontological models.

Finally, Table 2 presents the fulfilled requirements on research works presented in second section as an example per each KRi type. In this manner, it is possible to identify how recent research works have implemented KR, following the described requirements on this included in the third section.

4 Conclusions

The review of recent industrial automation research shows that there is a wide variety of distinct KRi for manufacturing systems. Moreover, the research on recent developments has motivated towards classification of them because differences between the implementation of KR in several works can be identified according to the concepts described in second section. Therefore, this classification can be used as a reference for KRi developers that need to set main functionalities of their development and some examples of recent implementations in the field. As it was described, the pre-

sented classification differentiates between three groups of KRi in the manufacturing systems domain: DSTi, CoTi and Full-KDi.

After reviewing the cited papers in this chapter, there is no doubt that KR can be beneficial for manufacturing systems because it is useful for supporting DST or, in case of Full-KDi, becoming the central component of a system. However, before designing any KRi, a set of requirements must be taken into account. Thus, the chapter presents a list of requirements that must be fulfilled, depending on the KRi type being developed.

It should be noted that the use of different languages and/or technologies in any development could modify requirements of implementations. However, this issue is avoided because the content of Table 1 is not dependent on technology. In other words, these requirements are conceptual so they can be satisfied through different choices.

References

Abburu, S. (2012). A Survey on Ontology Reasoners and Comparison. *International Journal of Computer Applications*, 57(17).

Asuncion Gomez-Perez, O. C. (2004). Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web (Advanced Information and Knowledge Pr.). http://www.biblio.com/book/ontological-engineering-examples-areas-knowledge-management/d/666682494. Accessed 2015.IV.01.

Brachman, R. J., & Levesque, H. J. (2004). Knowledge Representation and Reasoning. Morgan Kaufmann.

Dentler, K., Cornet, R., ten Teije, A., & de Keizer, N. (2011). Comparison of Reasoners for Large Ontologies in the OWL 2 EL Profile. *Semantic Web*, 2(2): 71–87. http://doi.org/10.3233/SW-2011-0034. Accessed 2015.IV.01.

Garetti, M., (2012). P-PSO Ontology for Manufacturing Systems. *Information Control Problems in Manufacturing, Vol.14/1*: 449–456. http://doi.org/10.3182/20120523-3-RO-2023.00222. Accessed 2015.IV.01.

Giovannini, A., Aubry, A., Panetto, H., Dassisti, M., & El Haouzi, H. (2012). Ontology-Based System for supporting Manufacturing Sustainability. *Annual Reviews in Control*, 36(2), 309–317. http://doi.org/10.1016/j.arcontrol.2012.09.012. Accessed 2015.IV.01.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2): 199–220. http://doi.org/10.1006/knac.1993.1008. Accessed 2015.IV.01.

ISA-95. ISA-95 standard for the integration of enterprise and control systems. http://www.isa-95.com/subpages/home/welcome.php. Accessed 2015.III.05.

Lastra, J. L. M., & Delamer, I. M. (2006). Semantic web services in factory automation: fundamental insights and research roadmap. *IEEE Transactions on Industrial Informatics*, 2(1): 1–11. http://doi.org/10.1109/TII.2005.862144. Accessed 2015.IV.01.

Lastra, J. L. M., Delamer, I. M., & Ubis, F. (2010). Domain Ontologies for Reasoning Machines in Factory Automation. ISA.

Maniraj, V., & Sivakumar, R. (2010). Ontology Languages – A Review. *International Journal of Computer Theory and Engineering*, 2(6): 887–891.

Noy, N. F., & McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. *Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880*.

http://liris.cnrs.fr/~amille/enseignements/Ecole_Centrale/What%20is%20an%20ontology%20and%20why%20we%20need%20it.htm. Accessed 2015.IV.01.

OWL (2004). Web Ontology Language Reference. http://www.w3.org/TR/owl-ref/. Accessed 2015.IV.01.

Puttonen, J., Lobov, A., & Lastra, J. L. M. (2013a). Maintaining a Dynamic View of Semantic Web Services Representing Factory Automation Systems, *IEEE 20th International Conference on Web Services (ICWS)*: 419–426. http://doi.org/10.1109/ICWS.2013.63. Accessed 2015.IV.01.

Puttonen, J., Lobov, A., & Lastra, J. L. M. (2013b). Semantics-Based Composition of Factory Automation Processes Encapsulated by Web Services, *IEEE Transactions on Industrial Informatics*, 9(4), 2349–2359. http://doi.org/10.1109/TII.2012.2220554. Accessed 2015.IV.01.

Puttonen, J., Lobov, A., & Lastra, J. L. M. (2014). On the Updating of Domain OWL Models at Runtime in Factory Automation Systems, *International Journal of Web Services Research*, 11(2): 46–66. http://doi.org/10.4018/ijwsr.2014040103. Accessed 2015.IV.01.

Ramis Ferrer, B. (2013). An ontological approach for modelling configuration of factory-wide data integration systems based on IEC-61499. Tampere University of Technology, Finland. https://dspace.cc.tut.fi/dpub/handle/123456789/21541. Accessed 2015.IV.01.

Ramis Ferrer, B., Gonzalez, L., Iarovyi, S., Lobov, A., Lastra, J. L.M., Vyatkin, V., & Dai, W. (2014). Knowledge-based web service integration for industrial automation, *12th IEEE International Conference on Industrial Informatics (INDIN)*: 733–739. http://doi.org/10.1109/INDIN.2014.6945604. Accessed 2015.IV.01.

RDF (2015). Semantic Web Standards. http://www.w3.org/RDF/. Accessed 2015.IV.01.

Russell, S. J., & Norvig, P. (2010). Artificial Intelligence: A Modern Approach. Prentice Hall.

SPARQL (2008). SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/. Accessed 2015.IV.01.

SPARQL 1.1 Update (2013). http://www.w3.org/TR/sparql11-update/. Accessed 2015.IV.01.

Stuckenschmidt, H., & Klein, M. (2003). Integrity and change in modular ontologies. Proceedings of the 18th International Joint Conference on Artificial Intelligence: 900–908, Acapulco, Mexico, 2003.

Uddin, M. K., Dvoryanchikova, A., Lobov, A., & Lastra, J. L. M. (2011). An ontology-based semantic foundation for flexible manufacturing systems. *37th IEEE Annual Conference on Industrial Electronics IECON 2011*: 340–345. http://doi.org/10.1109/IECON.2011.6119276. Accessed 2015.IV.01.

Zhang, Y., & Luo, X. (2014) An ontology-based knowledge model for engineering material selections. In: *Proceedings of the 2014 International Conference on Innovative Design and Manufacturing (ICIDM)* (pp. 53–58). http://doi.org/10.1109/IDAM.2014.6912670. Accessed 2015.IV.01.

Methodology to Develop Ontology for Manufacturing Systems

Anisha Sampath Kumar

Tampere University of Technology, Tampere, Finland

anisha.sampathkumar@tut.fi

Abstract. Manufacturing system ontology is a knowledge base for manufacturing systems. It is a formal representation of different aspects of the manufacturing domain. A knowledge driven approach for manufacturing system creates flexibility and capability for dynamic reconfiguration which makes the system adaptable to changes like addition of equipment, new tasks and changes in product and so on. Some of the contemporary methodologies for building ontology use Web Ontology Language (OWL). This chapter proposes a methodology, which extends such contemporary methodologies, with an aim to provide a standard approach for modelling the manufacturing system and supports the ontology developers to get a better ontology design. Moreover the chapter presents demonstration of the proposed methodology within a real use case.

1 Introduction

The competitiveness of manufacturing organizations depends on their ability to respond to changes timely and cost-effectively. Hence a manufacturing system must be capable of dynamically adapting its production in harmonization with the suppliers, customers, sub-systems within the firm and also with changes in the product being manufactured. A knowledge driven manufacturing system is flexible, re-configurable and adaptable to sudden changes in the system like introduction of new tasks, equipment, etc. This has directed the development of ontologies for manufacturing systems.

Ontologies allow formalizing representation of the knowledge based on a conceptualization. Manufacturing system ontology is an explicit specification of different concepts in manufacturing domain. These concepts include product, process, resources, etc. A methodology to develop manufacturing system ontology is discussed in this chapter. It provides a standard approach for knowledge acquisition in manufacturing domain, ontology building and SPARQL query template creation for manipulating the data in ontology. SPARQL Protocol and RDF Query Language (SPARQL) is a query language to retrieve and manipulate the data stored in RDF format.

The next section provides an overview of background of an existing methodology to develop ontologies. The proposed methodology is presented in third section. Fourth section demonstrates implementation of the discussed methodology within a real use case. The case study describes in detail the steps of the methodology.

2 Background

Some contemporary methodologies for developing ontology use Web Ontology Language Description Logic (OWL DL), i.e. OWL sublanguage. OWL is characterized by formal semantics and Resource Description Framework (RDF) based serialization for Semantic Web. OWL DL supports maximum expressiveness while retaining computational completeness and decidability. A standard methodology to develop knowledge base using OWL DL ontology language is discussed in Martinez Lastra et al. (2010). A brief overview of the steps involved is presented below.

Knowledge acquisition
Knowledge acquisition in the domain of interest is the first step for ontology building. The domain of interest must be thoroughly studied so that the different concepts in the domain can later be represented in the knowledge model.

Considering reuse of existing ontologies
There may be ontologies already existing in the domain of interest. If the concepts represented in those ontologies are similar to the concepts to be modelled then they can be reused instead of building from scratch.

Terminology selection
The next step is the terminology selection. The terms selected must be proper and linked to concept. The terminology must adhere to the standards provided by organizations like DIN, ISO, IEEE etc. The proper terms used in the domain can be found in the glossaries provided by respective Industrial associations. The handbooks and technical publications also provide the terminology. Thus proper terms can be chosen from various available sources as mentioned above and use it for ontology building. The use of synonyms must be avoided as it might lead to misleading of the concept.

Definition of Classes and their hierarchy
Classes provide an abstraction mechanism for grouping resources with similar characteristics. In OWL the top level classes are super classes and they have sub-classes which in turn can have more sub-classes and so on. The naming or terminology for class does not support the use of space character between words. For this purpose underline space"_" is used between words or CamelBackCapitalization is used. CamelBackCapitalization or Camel Case is the widely adopted writhing style.

Definition of properties
The properties provide the expressivity needed for the ontology model. There are two types of them. Object properties for establishing relationship between the classes and Datatype properties for establishing relationship between class and data types. The data types include Boolean, String, Integer, etc. The object property has Domain and Range. The individual of a class from Domain is linked to the individual of a class from Range using the object property. In case of data property, it links the individual of a class in Domain to a value of particular data type. The domain can contain multiple classes and can also be undefined. The properties can also be hierarchical with sub-properties. The hierarchical definition of properties adds more detail to the model.

Creation of additional properties
Additional properties are defined to establish relations between the existing one. They can define, if two properties are equivalent or inverse. They can also express logical capabilities of the property such as transitiveness, symmetry, functional inverse and functional.

Creation of Instances/Individuals
For every OWL class, a set of individuals called class extensions are defined. They are the instances of the class.

Creation of axioms/rules
The last step can be creation of axiom/rules to increase the expressivity of properties. SWRL is used to add more relations between classes. The use of SWRL by reasoners is still limited.

As mentioned above, *Knowledge acquisition* is the beginning of ontology building. The domain of interest should be studied to extract all the concepts and then formalize the knowledge acquired. This requires great deal of manpower and time.

Identifying key actors (either software or persons) in the system and their interactions are crucial for the ontology development. The traditional methodology does not define a standard approach for this. The new methodology discussed below provides a standard approach for knowledge acquisition, ontology model creation and template creation for queries that are required for the knowledge based system.

3 Methodology

The proposed methodology involves modelling of the manufacturing system in ontology format and also defines the techniques and the tools involved in the process. The methodology mainly focuses on ontology development using OWL DL language. The steps included in the methodology are: modelling the manufacturing system, creating templates for queries and creating templates for update queries. They are discussed in the following sub-sections.

Ontology Model
The creation of manufacturing system ontology model involves studying the manufacturing domain and representing the concepts in ontology.

Knowledge acquisition
As a first step towards formalization, the concepts have to be described in a systematic way inside their context unambiguously. For this purpose the Unified Modelling Language (UML) diagrams describing the manufacturing system are studied first. UML is a standard modelling language intended to be used for modelling business and similar processes. It is designed to provide a standard way to visualize the design of the system in diagrams.

The use case (UC) diagram is a UML diagram which designates the functional requirements and use goal for the system. It consists of internal or external agents for

making interactions with system, use cases and their relationships. A single UML diagram captures a particular functionality of a system. By studying all the UC diagrams which models the entire system, all the functional requirements can be gathered. It is a good approach for knowledge acquisition.

Building ontology

After the knowledge acquisition is complete, the next step is building the knowledge base for manufacturing system. The manufacturing system ontology is created using the OWL DL language as it supports maximum expressiveness. The ontology creation and manipulation is supported by ontology editor, which is an ontology development tool to reduce the complexity of the development process. There are different ontology editors in use like Protégé, SWOOP, Cerebra construct™, IBM IODT, KAON2, OntoTrack, OntoEdit and TopBraid Composer. They usually support one of available ontology languages.

The classes and their hierarchy in the ontology are defined based on the acquired knowledge. Different aspects of the system observed from UC are presented in a hierarchy of classes. Next the properties are defined to introduce the required expressivity for subsequent reasoning capability. The properties include the object property establishing relationship between classes and the data property forming relationship between classes and data types. After the classes and properties are defined, the required instances/individuals are created for the classes and property assertions are made.

Query templates

After the ontology model of manufacturing system is developed, we have to define how to store, update and retrieve the information represented in ontology. Queries are used to retrieve the information stored in data bases. Hence as a next step, templates are generated for queries. SPARQL Protocol and RDF Query Language (SPARQL) is used to create the queries. It is a query language recommended by World Wide Web Consortium (W3C) to retrieve and manipulate the data stored in RDF format.

Update Query templates.

Update queries are used to update the information stored in ontology. They are created using SPARQL/Update language for RDF graphs, which is also a W3C recommendation. The SPARQL/Update provides operations like updating the information, creating new instances and removing/deleting the information stored in ontology.

To define all the queries that are necessary for information handling in a manufacturing system, the flow of activities and actions within the manufacturing system should be known. This information can be obtained from UML activity diagram. The UML activity diagram shows a detailed execution of system by stepwise activities and actions with support for choice, iteration and concurrency. An activity represents a person or a software component performing some actions in the system. The actions are requesting data from system or sending data to system for updating. Hence, investigating the UML activity diagram is a useful approach to determine which queries are required. And then templates for queries and update queries are created based on it.

4 Case Study

The case study is based on the FluidHouse[1] assembly system. The step by step development of the ontology model for an assembly system, assuming implementation of the presented methodology, is explained in this section. The next subsection describes the realization and creation of the ontology model. Two consecutive subsections describe templates creation for queries and update queries for the Knowledge based system.

Creating Ontology Model.
Firstly, the ontology model is realized by studying the UML use case diagram shown below in Fig. 1. The use case diagram is analyzed to determine the functional requirements and use goal of the system. Based on the analysis, ontology model is created to meet the requirements. The ontology model created in this chapter is just one of the ways to represent the Manufacturing system based on knowledge acquired from UML diagrams.

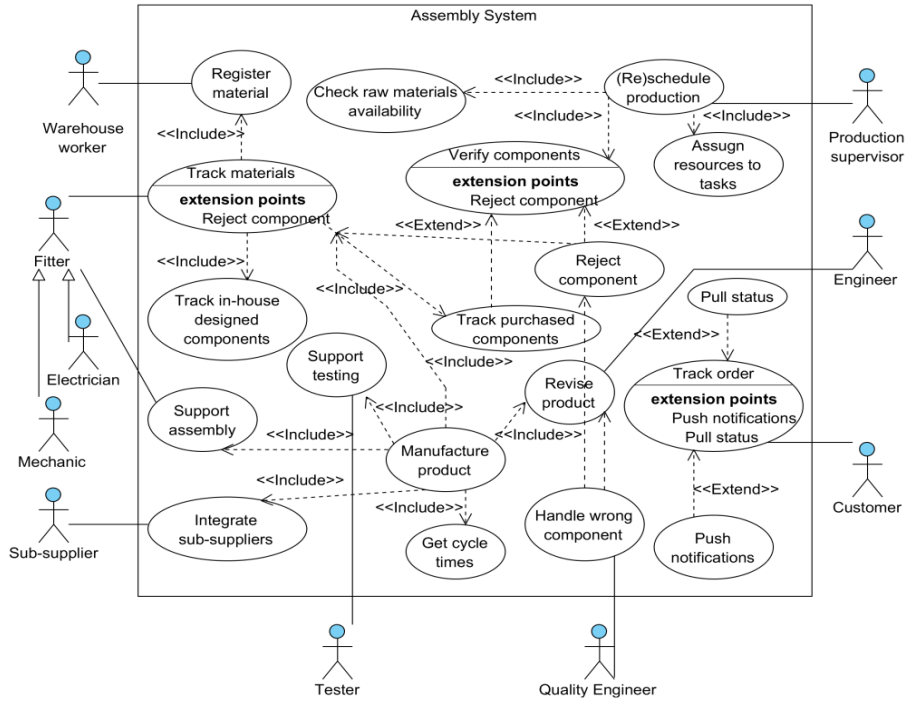


Fig. 1. FluidHouse Use Case Diagram

[1] FluidHouse is an engineering company specialized in hydraulic and oil lubrication automation. The link to Fluid House website: www.fluidhouse.fi

In the ontology model functional requirements derived from UML use case diagram are met by considering the MES functions of Product tracking, Resource allocation, Data acquisition, Labor Management and Scheduling, as described in (MESA. 1997a) and (MESA, 1997b). A separate class is created for these MES functions in the model with each function represented as an instance. The idea to represent functions of a system as a class was referred from (Long, 2008).

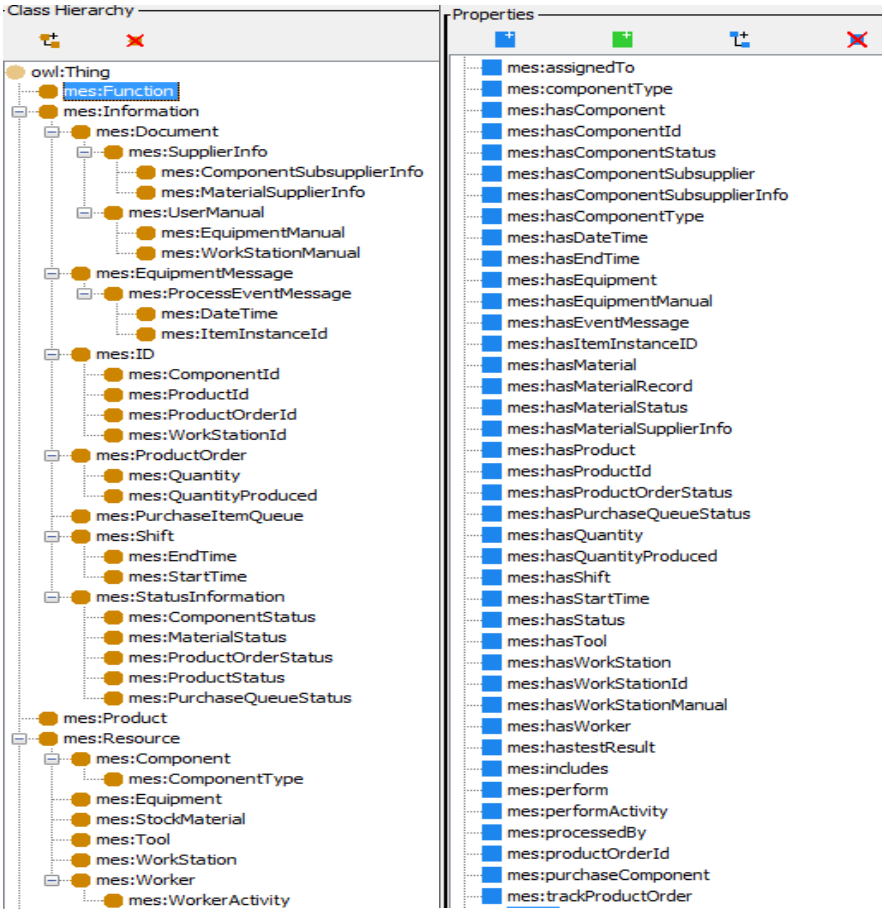


Fig. 2. MES ontology: **(a)** Class hierarchy; **(b)** Properties

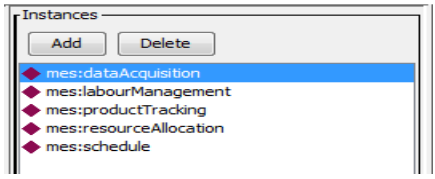


Fig. 3. Instances for class Function

The main classes in the ontology are Function, Information, Product and Resource. The class hierarchy is shown in Fig. 2 (a). The term ‘mes:’ is the namespace prefix for the ontology. The properties to establish relationships between classes, between classes and data types are defined. The properties defined are shown in Fig. 2 (b). The instances for class Function are shown in Fig. 3.

Creating templates for Query.

The activity diagram for the assembly system is shown in Fig. 4. From this diagram, the main activities are found as Designer, Purchase, Warehouse, Production Supervisor, Assembly, Outsourcing, Testing and Delivery.

The actions performed by the activities either changes some parameters in the system or require information about system to perform the action. This is supported by SPARQL update and SPARQL queries respectively.

The template for SPARQL update is discussed in the next section. The templates for queries are created with reference to (W3C, 2008).

The Production Supervisor may need to track the status of product order, products, individual parts and equipment working on the product with timing information. The query template for product tracking function is given in Fig. 5.

The Production Supervisor can also query for resource allocation using template shown in Fig. 6. The query returns how the equipment, workers etc. are allocated to each workstation and the various components and materials allocated for producing the product.

The query template for labor management is shown in Fig. 7. It retrieves the details of workers like which workstation they are assigned to, task, shift, work starting and ending time.

The stock materials and their status in warehouse is queried using template shown in Fig. 8.

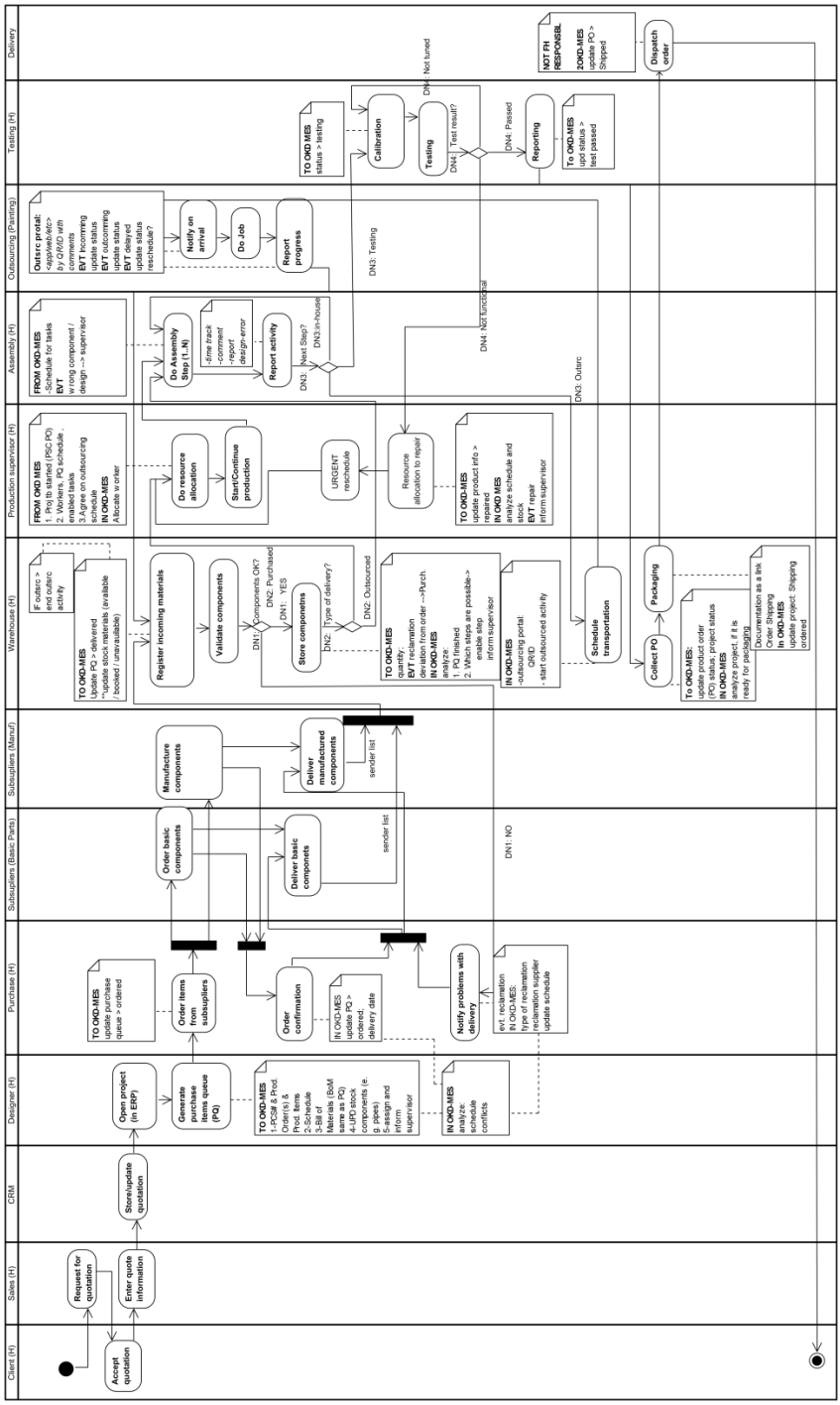


Fig. 4. FluidHouse Activity diagram

```
PREFIX mes:<http://www.escop-project.eu/OntologyMES.owl#>
SELECT ?Product ?ProductStatus ?Component ?ComponentStatus ?StockMaterial
?MaterialStatus ?WorkStation ?Equipment ?ProcessEventMessage ?DateTime
WHERE
 {  ?ProductTracking mes:trackProductOrder ?ProductOrder.
    ?ProductOrder mes:productOrderId ?productOrderId;
                  mes:hasProduct ?Product;
                  mes:hasQuantity ?Quantity;
                  mes:hasQuantityProduced ?QuantityProduced;
                  mes:hasProductOrderStatus ?ProductOrderStatus.
    ?Product mes:hasStatus ?ProductStatus;
          mes:hasComponent ?Component;
          mes:hasMaterial ?StockMaterial.
    ?Component  mes:hasComponentStatus ?ComponentStatus.
    ?StockMaterial mes:hasMaterialStatus ?MaterialStatus.
    ?ProductStatus mes:processedBy ?WorkStation.
    ?WorkStation mes:hasEquipment ?Equipment.
    ?Equipment mes:hasEventMessage ?ProcessEventMessage.
    ?ProcessEventMessage mes:hasDateTime ?DateTime.
 }
```

Fig. 5. Query for Product Tracking

```
PREFIX mes:<http://www.escop-project.eu/OntologyMES.owl#>
SELECT ?Worker ?Shift ?WorkerActivity ?WorkStationId
?StartTime ?EndTime
WHERE
  {
   ?LabourManagement mes:hasWorker ?Worker.
   ?Worker mes:performActivity ?WorkerActivity;
           mes:assignedTo ?WorkStationId;
           mes:hasShift ?Shift.
   ?Shift mes:hasStartTime ?StartTime;
          mes:hasEndTime ?EndTime.
  }ORDER BY ?Worker
```

Fig. 6. Query for Labour Management

```
PREFIX mes:<http://www.escop-project.eu/OntologyMES.owl#>
SELECT ?WorkStation ?WorkStationId ?Equipment ?Worker ?Product ?StockMaterial
?Component
WHERE
 { ?ResourceAllocation mes:hasWorkStation ?WorkStation.
  ?WorkStation mes:hasWorkStationId ?WorkStationId.
  ?WorkStation mes:hasEquipment ?Equipment.
  ?WorkStation mes:hasWorker ?Worker.
  ?ResourceAllocation mes:hasProduct ?Product.
  ?Product mes:hasComponent ?Component;
        mes:hasMaterial ?StockMaterial.
 }
```

Fig. 7. Query for Resource allocation

```
PREFIX mes:<http://www.escop-project.eu/OntologyMES.owl#>
SELECT ?StockMatrial ?MaterialStatus
WHERE
 {
  ?StockMaterial mes:hasMaterialStatus ?MaterialStatus;
            mes:hasMaterialRecord ?MaterialSupplierInfo.
 }
```

Fig. 8. Query for Stock materials

Creating templates for update query.

The UPDATE query is used to modify the data of the model. The template for UPDATE queries are created with reference to (W3C, 2013). The variable within <> in the following queries are to be replaced accordingly. The prefix 'mso:' must be added in case of instance of a class.

In Designer, the product order is updated. The query template to update the product order is shown in Fig. 9.

The next activity is Purchase. Query to update purchase related information, like purchase items, sub-suppliers and purchase queue status is given by Fig. 10.

The incoming material to warehouse can be a purchased component from sub-supplier or outsourced component for painting. So the query to update the type of incoming component and their status is shown in Fig. 11. The query to update the status of stock materials in warehouse is shown in Fig. 12.

The Production supervisor performs resource allocation and the template to update the allocated resources is given in Fig. 13.

The update query for assembly involves updating status of each part, which is in processing and timing for each step. It is shown in Fig. 14.

The product status is updated using the query given in Fig. 15. <status> can be for e.g."inProcess" if the product is being processed in Assembly or e.g."testing" during Testing activity.

The update query template for product order status is given in Fig. 16 and <product order status> is also replaced based on the activity.

```
PREFIX mes:<http://www.escop-project.eu/OntologyMES.owl#>
DELETE { ?ProductOrder mes:productOrderId ?productOrderId;
                mes:hasProduct ?Product;
                mes:hasQuantity ?Quantity;
                mes:hasQuantityProduced ?QuantityProduced;
                mes:hasProductOrderStatus ?ProductOrderStatus.
    }
INSERT { ?ProductOrder mes:productOrderId <product Order id>;
                mes:hasProduct <product>;
                mes:hasQuantity <quantity>;
                mes:hasQuantityProduced <quantity produced>;
                mes:hasProductOrderStatus <product order status>;
WHERE
  { ?ProductOrder mes:productOrderId ?productOrderId;
                mes:hasProduct ?Product;
                mes:hasQuantity ?Quantity;
                mes:hasQuantityProduced ?QuantityProduced;
                mes:hasProductOrderStatus ?ProductOrderStatus.
    }
```

Fig. 9. Update query for Product order

```
PREFIX mes:<http://www.escop-project.eu/OntologyMES.owl#>
DELETE { ?PurchaseItemQueue mes:hasPurchaseQueueStatus ?PurchaseQueueStatus;
                mes:hasComponentStatus ?ComponentStatus.
    }
INSERT { ?PurchaseItemQueue mes:hasPurchaseQueueStatus <purchase queue
status>;
                mes:hasComponentStatus <component status>.
    }
WHERE { ?PurchaseItemQueue mes:hasPurchaseQueueStatus ?PurchaseQueueStatus;
                mes:hasComponentStatus ?ComponentStatus.
    }
```

Fig. 10. Update query for Purchase

```
PREFIX mes:<http://www.escop-project.eu/OntologyMES.owl#>
DELETE { ?Component mes:hasComponentStatus ?ComponentStatus;
                mes:hasComponentType ?ComponentType.
    }
INSERT { <component> mes:hasComponentStatus <component status>;
                mes:hasComponentType <component type>.
    }
WHERE { ?Component mes:hasComponentStatus ?ComponentStatus;
                mes:hasComponentType ?ComponentType.
    }
```

Fig. 11. Update query for Incoming materials in Warehouse

```
PREFIX mes:<http://www.escop-project.eu/OntologyMES.owl#>
DELETE { ?Stockmaterial mes:hasMaterialStatus ?MaterialStatus.}
INSERT { <stock material> mes:hasMaterialStatus <material status>.}
WHERE { ?StockMaterial mes:hasMaterialStatus ?MaterialStatus.}
```

Fig. 12. Update query for Stock materials

```
PREFIX mes:<http://www.escop-project.eu/OntologyMES.owl#>
DELETE { ?WorkStation mes:hasWorker ?Worker.
        ?ResourceAllocation mes:hasProduct ?Product.
        ?Product mes:hasComponent ?Component;
                mes:hasMaterial ?Material.
      }
INSERT {
        mes:AssemblyStation mes:hasWorker <worker>.
        mes:TestingStation mes:hasWorker <worker>.
        ?ResourceAllocation mes:hasProduct <product>.
        ?Product mes:hasComponent <component>;
                mes:hasMaterial <material>.
      }
WHERE { ?ResourceAllocation mes:hasWorkStation ?WorkStation.
       ?WorkStation mes:hasWorkStationId ?WorkStationId;
                  mes:hasEquipment ?Equipment;
                  mes:hasWorker ?Worker.
       ?ResourceAllocation mes:hasProduct ?Product.
       ?Product mes:hasComponent ?Component;
              mes:hasMaterial ?StockMaterial.
     }
```

Fig. 13. Update query Resource allocation

```
PREFIX mes:<http://www.escop-project.eu/OntologyMES.owl#>
DELETE {
      <product> mes:hasStatus ?ProductStatus.
      ?Component  mes:hasComponentStatus ?ComponentStatus.
      ?StockMaterial mes:hasMaterialStatus ?MaterialStatus.
      ?Equipment mes:hasEventMessage ?ProcessEventMessage.
      mes:itemWorkStart mes:hasDateTime ?DateTime.
    }
INSERT {
      <product> mes:hasStatus <status>.
      ?Component mes:hasComponentStatus <component status>.
      ?StockMaterial mes:hasMaterialStatus <material status>.
      ?Equipment mes:hasEventMessage <event message>.
      mes:itemWorkStart mes:hasDateTime <date time>.
    }
WHERE {
      ?Product mes:hasStatus ?ProductStatus;
            mes:hasComponent ?Component;
            mes:hasMaterial ?StockMaterial.
      ?Component  mes:hasComponentStatus ?ComponentStatus.
      ?StockMaterial mes:hasMaterialStatus ?MaterialStatus.
      ?ProductStatus mes:processedBy ?WorkStation.
      ?WorkStation mes:hasEquipment ?Equipment.
      ?Equipment mes:hasEventMessage ?ProcessEventMessage.
      ?ProcessEventMessage mes:hasDateTime ?DateTime.
    }
```

Fig. 14. Update query for Assembly

```
PREFIX mes:<http://www.escop-project.eu/OntologyMES.owl#>
DELETE { <product> mes:hasStatus ?ProductStatus.}
INSERT { <product> mes:hasStatus <status>.}
WHERE { ?Product mes:hasStatus ?ProductStatus.}
```

Fig. 15. Update query for Product status

```
PREFIX mes:<http://www.escop-project.eu/OntologyMES.owl#>
DELETE { ?ProductOrder mes:hasProductOrderStatus ?ProductOrderStatus.}
INSERT { ?ProductOrder mes:hasProductOrderStatus <product order status>.}
WHERE { ?ProductOrder mes:hasProductOrderStatus ?ProductOrderStatus.}
```

Fig. 16. Update query for Product order status

5 Conclusions and Summary

In this chapter new methodology is presented for development of ontologies for Manufacturing Systems. The existing methodologies do not define any standard approach for knowledge acquisition and formalization. It takes a great deal of time to

acquire knowledge in the domain throughly and build a new ontology without a proper procedure.

The proposed methodology extends the basic steps used in building ontologies using OWL. It makes use of UML diagrams for knowledge acquisition and developing ontology model. The UML use case diagram helps to understand the requirements and goal of the system. Knowing the different concepts involved in the system, it results in efficient modelling. The queries that are required to manipulate the data in the ontology are identified by analyzing the UML activity diagram since it defines the activities and interactions between the various modules of the system. Thus the proposed methodology helps ontology builders to create efficient models following a standard approach with reduced effort and time.

References

Long, W. (2008). Construct MES ontology with OWL. *ISECS International Colloquim on Computing, Communication, Control and Management*: 614-617.

MESA (1997a). MES Functionalities & MRP to MES Data Flow Possiblities, White Paper2, *Manufacturing Execution Systems Association*, Pittsburgh.

MESA (1997b). MES explained: A high level vision. White Paper6, *Manufacturing Execution Systems Association*, Pittsburgh.

W3C (2008). SPARQL 1.1 Query Language, World Wide Web Consortium. http://www.w3.org/TR/rdf-sparql-query/. Accessed 2014.XI.25.

W3C (2004). OWL Web Ontology Language. World Wide Web Consortium. http://www.w3.org/TR/owl-ref/. Accessed 2014.XI.25.

W3C (2013). SPARQL 1.1 Update. World Wide Web Consortium. http://www.w3.org/TR/sparql11-update/. Accessed 2014.XI.25.

Martinez Lastra, J.L., Delamer, I.M., and Ubis, F. (2010). Domain Ontologies for Reasoning Machines in Factory Automation. *International Society of Automation*.

Ontology-Based Backend Engine for Manufacturing and Logistics Systems

Xiangbin Xu

Tampere University of Technology, Tampere, Finland

xiangbin.xu@tut.fi

Abstract. Manufacturing system ontology (MSO) is a generic semantic model for Open Knowledge-driven Manufacturing Execution Systems (OKD-MES). It serves not only as a database, but also as a knowledge base to represent the concepts in the domain of manufacturing. The data relevant to manufacturing activities are then be mapped to MSO. The aim of MSO is to systematically share manufacturing knowledge for the development of MES in various industries although manufacturing and logistics systems are the focus of this chapter. Hence an ontology-based backend engine (OBBE) is needed to implement and functionalize MSO in order to collect data, query the data in MSO and communicate with other modules in OKD-MES. This chapter proposes the functionalities and architecture for OBBE that interacts with other modules in OKD-MES and introduces the implementation of OBBE.

1 Introduction

According to the architecture of Open Knowledge-driven Manufacturing Execution Systems, an ontology-based backend engine provides services to interact with: (i) Orchestration Layer, i.e. a module that automatically arranges, coordinates, and manages complex services; (ii) Physical Layer, i.e. a distributed module that publishes device events and receives operation requests; (iii) Visualization Layer, i.e. a module that generates dynamic graphical visualization for the end users. In addition, OBBE is supposed to provide a container for MSO and to be able to execute CRUD (Create, Read, Update and Delete) operations for the data in MSO. Data security, integrity and concurrency issues should be considered by OBBE. Furthermore, specifically for manufacturing and logistics domain, concise functions and queries should be prepared for retrieving relevant data from MSO. Based on aforementioned requirements, the architecture and functionalities of OBBE are designed. RESTful web services are selected as fundamental blocks of OBBE because RESTful resource-oriented services are simpler, easier to use, more interoperable, and easier to combine than RPC-style services (Leonard and Sam, 2007). Several web application platforms that implement RESTful web services are available for OBBE, such as Apache CXF, Spring, Node.js and Vert.x. In terms of container of MSO, OBBE utilizes RDF triple store for persistence of MSO data.

2 System requirements

2.1 Non-functional requirements

Based on the role of OBBE in the architecture of OKD-MES, its non-functional requirements are described in Table 1. Non-functional requirements specify some fundamental features of OBBE.

Table 1. Non-functional requirements

No.	Requirement	Description
1	Accessibility	The services of OBBE must be accessible through Internet.
2	Deployment	OBBE should be deployable independently.
3	Extensibility	OBBE should be extendable such as adding new services.
4	Open source	OBBE should be open source.
5	Platform	OBBE should be cross-platform.
6	Scalability	OBBE should be able to handle large concurrent requests.
7	Security	OBBE should be immutable to bad requests.

2.2 Functional requirements

The core role of the OBBE is to provide MSO related services, namely populate MSO, update information in it and query information against. Hence four major requirements are needed for providing MSO services as shown in Table 2.

Table 2. MSO services requirements

No.	Requirement	Weight
1	Capability for persistent storage of MSO	Must
2	Capability to import/export MSO in OWL file	Must
3	Capability of querying information against MSO	Must
3.1	Availability of SPARQL engine	Must
3.2	Implementation of SPARQL querying methods	Must
3.3	Capability of handling exceptions in query format	Must
4	Security features to allow protection of models	Should
4.1	Protection from unauthorized queries	Could
4.2	Protection from harmful queries	Could

The MSO contains generic concepts of manufacturing systems in format of OWL file. It is necessary for the OBBE to integrate a graph database for persistent storage so that updated data can be populated in MSO. Obviously the OWL file should be able to be imported/exported to/from this database. The graph database also should provide SPARQL engine which is essential for updating and retrieving data by using SPARQL query. Several available query engines may be found in (W3C, 2013).

Once the graph database is selected the backend engine should implement SPARQL querying methods and query exception handler to integrate the database. In addition, data security is also a concern. The backend engine should protect MSO from unauthorized and harmful queries.

Apart from MSO services, services of the backend engine will be organized by service orchestration of OKD-MES. Orchestration refers to an executable business process that can interact with both internal and external Web services. The interactions occur at the message level. They include business logic and task execution order, and they can span applications and organizations to define a long-lived, transactional, multistep process model (Peltz, 2003). Service orchestration is the coordination and arrangement of multiple services exposed as a single aggregate service. Service orchestration in the eScop architecture implements two main functions. First, it composes the services based on matching services required by process definition and the services provided by system components. Secondly, it executes the composed services corresponding to the process and providing related information to the OKD-MES. Functional requirements for the backend engine considering the two named functions of service orchestration will be specified.

The manufacturing process should be predefined before service orchestration composes complex service. The process definition describes a sequence of operations to be performed in order to produce a product. The operations may have dependency on required conditions. Both operations and conditions should be represented as concepts in MSO so service orchestration could map them with their real implementations by using services of the backend engine.

As in case of redundancy of operations in the system the service composer should be able to distinguish different implementations of the operation and to support decision about which one is to be employed; the MSO should also include other details about service. Reasonable minimal level of description for manufacturing systems should include two levels of such descriptions: namely, the device level and operation level. These levels allow decoupling of technical details about the device from functional description of the operation. Device description may include details about location of device, its power consumption, its schedule etc. Operation description besides functional description of the operation should provide information about parameters, quality, cost of operations and so on. This information may be employed for decision support not only on composition but also on execution.

For orchestration of service compositions, on the one hand information stored in MSO should be accessible; on the other hand OBBE should be capable to update the knowledge model depending on orchestration process. The information which should be placed in MSO by service orchestration may be order and product related. In addition to this information, description of equipment status and states should be provided by OBBE to ORL on demand. Last requirement is related to representation of physical layer devices and their status, which in turn requires some mechanism for subscription on OBBE side as well. The requirements for representation layer from orchestration framework are presented in Table 3.

Table 3. MSO services requirements from the Service Orchestration

No.	Requirement	Weight
1	Provides ORL access to read and update the knowledge model	Must
2	Provides ORL with access to process description	Must
3	Provides ORL with access to operations	Must
4	Provides ORL with access to system status	Must
5	Provides ORL with access to update process execution related entities	Must

Besides, Visualization Layer (VIS) will be the final border between internal elements and physical end user of the system. OBBE also serves as a bridge between visualization and run-time data. VIS uses HTML5/SVG technology to visualize the graphics, dashboards, trends, and reports, bringing the visualization to any smartphone, tablet, web browser or web-enabled device. The interactions among VIS and OBBE are:

- Requests the visualization info on OBBE
- Gets the ontology model (objects, relationships, list of data sources, etc.)
- Generates the visualization display from the ontology model
- Responds to changes in ontology model (subscription mechanism)

Hence the requirements from VIS can be summarised as shown in Table 4.

Table 4. MSO services requirements from the Visualization layer

No.	Requirement	Weight
1	Provides VIS access to read visualization information from the knowledge model	Must
2	Translate visualization information to standard graphic representation messages	Must

Finally, OBBE has to be able to interact with eScop devices through device discovery mechanism. It means OBBE will obtain device description from the device and register the device in MSO. Therefore the functional requirements from device level are described in Table 5.

Table 5. MSO services requirements from the Device layer

No.	Requirement	Weight
1	Obtains device description from device URL.	Must
2	Inserts device description in MSO.	Must
3	Deletes device description in MSO when the device is not available anymore.	Must

3 System architecture

The architecture of OBBE is shown in Fig. 1.The services provided by OBBE can be classified to four modules: Device registration module, Visualization module, Service Handler module, and Manager Module. The modules interact with the external components of eScop architecture.

The Device Registration Module interacts with the physical layer devices; The Visualization Definer Module interacts with the Visualization layer agent; The Service Handler module works with the Orchestration layer and to the order input system (in the picture, Production Manager) and Manager Module is designed for System Administrator.

The details of the modules are described in Table 6.

Table 6. Modules of the OBBE

Module	Description
Device Registration Module	A service module to register or unregister eScop devices in the MSO (Manufacturing System Ontology) through device catalogue service. Provides mechanism of receiving new devices and detecting devices that become unavailable. Enables access to device existence and their capabilities in the network. Mainly works with Physical aspects in MSO.
Visualization Provider Module	A service module to retrieve, update the visualization information, for example GUI components and their properties, in MSO according to the query from Visualization Agent. Mainly interacts with Visualization, Technological and Physical aspects of MSO.
Service Handler Module	A service module to communicate with Service Composer, Orchestrator Services and Production Manager. The module should expose services to provide information requested from the Service composer, receive update requests from Orchestration service and receive update requests from Production Manager. The module must handle description of the services, configuration and status of the system, SPARQL over HTTP could be used for flexibility of the approach, but it is worth to mention it is necessary to introduce authorization and verification mechanism for SPARQL over HTTP in order to keep data integrity of the MSO. Besides, SPARQL query API can also be used. All aspects of MSO with possible exception of Visualization domain are important for this module.
Manager Module	A service module to provide configuration tools for system administrator to maintain, configure OBBE and manipulate the data in MSO. It also provides an web interface for viewing the Ontology data in MSO.
Internal functions	Internal functions are the common functions shared by the service modules. SPARQL Query Factory provides all the SPARQL queries for information retrieval and updates. Ontology Connector serves to send SPARQL query to MSO server (SPARQL over HTTP or SPARQL query api).

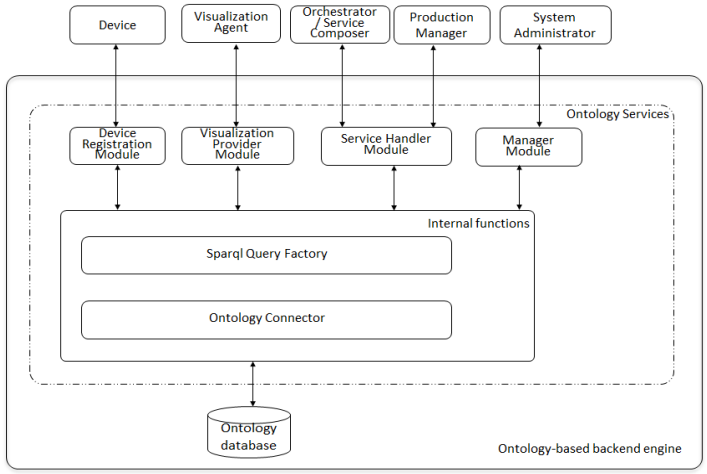


Fig. 1. Architecture of OBBE

4 System implementation

4.1 Development environment and framework

The backend engine is a Java 1.8 Maven project developed on Netbeans IDE. It is based on Spring Framework. The Spring Framework is an open source application framework and inversion of control container for the Java platform. It was created by Johnson et al. (2014) and first released under the Apache 2.0 license in June 2003.

The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform. It is selected not only because it provides RESTful web service framework but also other useful features such as Dependency Injection. As of writing this chapter, the current version is Spring Framework 4.0, which was released in December 2013.

Notable improvements in Spring 4.0 include support for Java SE 8, some aspects of Java EE7, WebSocket and simplicity for developing non-blocking RESTful services.

Spring Boot is convention-over-configuration solution for creating stand alone, production-grade Spring based Applications that can be simply run. It takes an opinionated view of the Spring platform and third-party libraries so the application developer can get started with minimum fuss (Phillip et al., 2014). It provides several advantages:

- Create stand-alone Spring applications
- Provide a range of non-functional features that are common to large classes of projects (e.g. embedded servers, security, metrics, health checks, externalized configuration).
- Absolutely no code generation and no requirement for XML configuration.

Due to the use of Spring Boot, the backend engine can be compiled as a single JAR file. Spring Boot includes a number of additional features to help developers monitor and manage their applications when it's pushed to production.

One of those features is Spring Boot Actuator. With it enabled it is possible to manage and monitor the application using HTTP endpoints, with JMX or even by remote shell (SSH or Telnet). Auditing, health and metrics gathering can be automatically applied to the application.

With the help of Spring Boot, the project configuration is significantly simplified. The following explanation of the project POM configuration shows the key dependency of the backend engine.

The project inherits from the spring-boot-starter-parent project to obtain sensible defaults:

```
<parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.1.8.RELEASE</version>
</parent>
```

The project uses the Apache HttpComponent project which is focused on HTTP and associated protocols:

```
<dependency>
<groupId>org.apache.httpcomponents</groupId>
<artifactId>httpcore</artifactId>
<version>4.3.2</version>
<type>jar</type>
</dependency>
<dependency>
<groupId>org.apache.httpcomponents</groupId>
<artifactId>httpclient</artifactId>
<version>4.3.5</version>
<type>jar</type>
</dependency>
```

The project uses Jena ARQ and Core API:

```
<dependency>
<groupId>org.apache.jena</groupId>
<artifactId>jena-arq</artifactId>
<version>2.12.1</version>
</dependency>
<dependency>
<groupId>org.apache.jena</groupId>
<artifactId>jena-core</artifactId>
<version>2.12.1</version>
<type>jar</type>
</dependency>
```

4.2 Graph database for MSO

Fuseki server is used as a SPARQL server that provides REST-style SPARQL HTTP Update, SPARQL Query, and SPARQL Update using the SPARQL protocol over HTTP. It includes a built-in version of TDB (a component of Jena for RDF storage and query) to persist Ontology model. Three service endpoints are provided:

- SPARQL query: http://*host*/dataset/query
- SPARQL update: http://*host*/dataset /update
- SPARQL Http update: http://*host*/dataset/date
- SPARQL file upload: http://*host*/dataset /upload

Where dataset is a URI path, the default host is localhost:3030. The first three service endpoints can also be accessed by using a query engine called Jena ARQ API. This simplifies the integration of the graph database with OBBE because in OBBE Jena ARQ can be used to execute SPARQL queries against MSO. It is worth to mention that Fuseki itself does not offer security and access control although it can be set to only listen to local host. Hence OBBE needs to implement security and access control to protect MSO from unauthorized requests.

4.3 Non-blocking RESTful web services

The RESTful web services provided by the OBBE are designed according to the functional requirements defined in Section 2.2. This section explains the services that are implemented in the four modules defined in the third section. In addition, the RESTful services are designed in a non-blocking style, i.e. without allocating a thread to each request, in order to be able to handle large number of concurrent requests in the application servers. This is important because of the role of the backend engine in the eScop architecture. It has to interact with various other layers such as the device layer, the orchestration layer, and particularly the visualization layer which may have a large number of connected users concurrently. Hence it can cause the blocking of the REST web service.

The request thread will be locked during the processing of this method. If the method needs to make a long running call to an external resource, such as another REST or SOAP service, or a database, the request thread will be blocked during the waiting for the external resource to respond. Thus, to solve this limit of blocking REST service, the REST services of OBBE does not return the actual result to the container servlet, but instead an object called a DeferredResult, that will receive the result at some time in the future. The result will be filled in by some other thread, typically using a callback-object.

The Spring framework will hand over the result to the container servlet that sends it back to the client. In this case a long running call to an external resource can take place without blocking the request thread. In this way, the scalability of OBBE is improved. The key services of the OBBE are explained in the following subsections.

Device Registration Module.
Device Registration Module contains the services for registering and removing eScop device information in MSO. Fig. 2 demonstrates the interaction between OBBE and eScop devices. Each eScop device will join in the multicast group when it starts up. OBBE sends UDP datagram packet containing device registration URL to the multicast group. Each device in the group will receive the broadcasting message and send Hello message to the device registration service.

The Hello message is produced in the JSON format {“id”:”RTU1”,”url”:” http://localhost:8083/”}. Following the URL contained in the Hello message, OBBE retrieves the device description as an OWL file and inserts the device information into the MSO.

The services implemented in this module are described in Table 7.

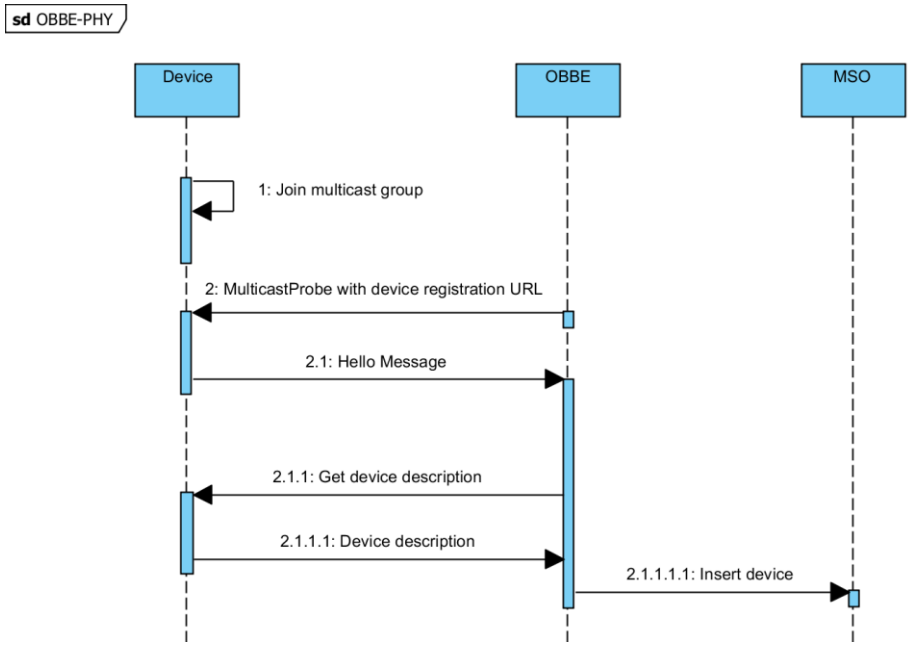


Fig. 2. Interaction between OBBE and eScop device

Table 7. Example of service endpoints

Service route	GET	POST	PUT	DELETE
/RTU	√	√	X	X
/RTU/{id}	√	X	√	√
/RTU/{id]/device	√	√	X	X
/RTU/{id]/device/{id}	√	X	√	√

The Device Registration module publishes two kinds of resources: RTUs (eScop remote terminal units) and devices (physical devices such as conveyor controlled by a given RTU). The service route /RTU is explained by an example. The root /RTU resource is used to retrieve (with GET) the list of all available RTUs:

=> GET /RTU
Accept:application/json
<= 200 OK
Content-Type:application/json

or

<= 503 Connection to MSO failed (When failing to connect to the graph database)

The root /RTU resource is used to add (with POST) new eScop RTU in the eScop system:

=> POST
Content-Type:application/json
RequestBoday:{"id":"RTU1","url":"http://localhost:8083/"}
<= 201 Created
Location:http://localhost:9000/RTU/id

or

<= 503 Connection to MSO failed

or

<= 409 RTU already exists in MSO

or

<= 400 Cannot retrieve RTU info from its url

ServiceHandler Module.

The services in Service Handler Module provide necessary information for service orchestrator to orchestrator services. Example services are presented in Table 8.

The /DiscoveredDevice route (with GET) lists all the registered devices in MSO. The /DiscoveredDevice/{deviceId}/event/{eventId} route (with GET) returns the device event URL according to the parameter deviceId and eventId. The schema of the JSON message returned by this service is:

{
eventId (string): unique identifier for event from eScop device,
url (string): url of the event
}

Table 8. Example of service endpoints

Service route	GET	POST	PUT	DELETE
/DiscoveredDevice	√	X	X	X
/DiscoveredDevice/{deviceId}/event/{eventId}	√	X	X	X
/DiscoveredDevice/{deviceId}/operation/{operationId}	√	X	X	X

The /DiscoveredDevice/{deviceId}/operation/{operationId} route (with GET) returns the device operation URL according to the parameter deviceId and operationId. The schema of the JSON message returned by this service is

```
{
operationId (string): unique identifier for operationin eScop device,
url (string): url of the operation
}
```

Visualization Provider Module

Services in Visualization Provider Module provide knowledge of visualization displays to Visualization agent which is a commercial software ICONICS MobileHMI (ICONICS, 2013). This software is a web solution for creating rich, real-time universal visualizations for mobile dashboards for energy, quality or production metrics in real-time. Visualization displays are dynamically composed based on the visualization symbols and symbol library storage using the services in the Visualization Provider Module. The interaction between Visualization agent and OBBE includes two phases of the interaction: configuration phase and runtime phase. Table 9 shows the service routes in this module.

Table 9. Example of service endpoints

Service route	GET	POST	PUT	DELETE
/Vis/{elementId}	√	X	X	X
/Vis/configure/{elementId}	√	√	X	X

The /Vis/{elementId} route (with GET) provides the visualization information ofthe given element. The schema of the JSON message returned by this service is

```
{
  metadata (string):null,
  containerType (string): graphic container type,
  id (string): element id,
  type (string): element type,
  visTemplate (string): visualization template name,
  children (array): child elements
{
{CHILD_ELEMENT_URL} (array):{
       x (float):x coordinate,
       y (float):y coordinate
    },
{CHILD_ELEMENT_URL} (array):{
       x (float):x coordinate,
       y (float):y coordinate
  },
  parameter (string):parameter
}
```

/Vis/configure/{elementId}route (with GET) offers a configuration interface for the user to define the layout information of the graphic elements as shown in Fig. 3.

ELEMENT	X	Y
Vis_Reservoir_Module	91.27420127899649	233.68934867265364
Vis_Pumping_Module	255.72565619394015	234.14637537021207
Vis_Filtering_Module	418.76934318063985	236.46612527608664
Vis_Cooling_Module	574.9349976058136	235.72813956969264
Vis_MS1	702.3713257574934	236.128403743754

Fig. 3. Visualization configuration web interface of OBBE

Once the user completes the configuration, the interface will send HTTP POST message to the service /Vis/configure/{elementId} so that the layout information will be updated in MSO.

Manager Module.

Manager Module provides tools for the system administrator to configure and maintain OBBE. One important tool is the administration web interface for viewing the MSO contained in the Fuseki server. Its URL is /Manager/Administrator. It allows the administrator dynamically to check the data in MSO at run time.

As shown in Fig. 4, the interface has four views namely Active Ontology, Class view, Property view and Individual view. These four views give comprehensive information about the data of MSO. Moreover, it provides interface to directly manipulate the data of MSO by using SPARQL query and also to upload OWL file into the Fuseki server.

Before using the interface, the administrator has to connect the interface to the Fuseki server by entering correct port number in “DATASET” menu. If the port number is correct, the interface shows “connected successfully”. It is worth to mention that the administration tool is designed for the user familiar with the OBBE and the MSO as far, as the data integrity is concerned.

Fig. 4. Administration web interface of OBBE

5 Conclusions

In this chapter an ontology-based backend engine is introduced. The design of the engine follows the specification of functional and non-functional requirements from other components with which it has to interact. Due to the need of ontology knowledge base, the engine integrates Fuseki server to provide CRUD operations on the data in the manufacturing system ontology model. Finally, its development framework and implementation are explained.

The backend engine developed in the project demonstrates a way of how to serve information contained in ontology knowledge base through RESTful web services.

References

ICONICS (2013), MobileHMI version 10.81. http://www.iconics.com/Home/Products/Web-Solutions/MobileHMI.aspx. Accessed 2014.XII.14.

Johnson R., Höller, J., Donald, K., Sampaleanu, C., Harrop, R., Risberg, T., Davison, D., Kopylenko, D., Pollack, M., Templier, T., Vervaet, E., Tung, P., Hale, B., Colyer, A., Lewis, J., Leau, C., Fisher, M., Brannen, S., Laddad, R., Poutsma, A., Beams, C., Abedrabbo, T., Clement, A., Syer, D., Gierke, O., Stoyanchev, R., Webb, P., Winch, R., Clozel, B., Nicoll, S., & Deleuze, S. (2014). Spring Framework Reference Documentation. http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/. Accessed 2014.XI.15.

Leonard, R., & Sam, R. (2007). RESTful Web Services. O'Reilly, p 221.

Peltz, C. (2003). Web Services Orchestration and Choreography. *Computer* 36(10): 46-52.

Phillip, W., Dave, S., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C., & Deleuze, S. (2014). Spring Boot Reference Guide. http://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/. Accessed 2014.XI.15.

W3C (2013). SPARQL Implementations. http://www.w3.org/wiki/SparqlImplementations#Query_Engines. Accessed 2014.XI.15.

Ontology-Driven Visualization for Manufacturing Systems

Anisha Sampath Kumar

Tampere University of Technology, Tampere, Finland

anisha.sampathkumar@tut.fi

Abstract. This chapter proposes a representation of knowledge required for visualization of manufacturing systems in ontology. The Visualization layer (VIS) of the eScop architecture represents the front-end of the Open, Knowledge-driven Manufacturing Execution Systems (OKD-MES). VIS generates dynamic graphical visualization of the system and makes it available on any standard computer, smartphone, tablet or web-enabled devices. The information required to generate visualization should somehow be stored and presented to VIS. Manufacturing System Ontology (MSO) serves this purpose. MSO is a knowledge base for OKD-MES that stores knowledge on different aspects of the system. By representing the knowledge for visualization also in the ontology, the visualization information will be stored in the same format as any other information about given system. This simplifies the system management. The chapter discusses how the actual component in a system and its graphic for visualization can be linked through ontology. The representation of different levels of details about the graphics in the ontology is also proposed.

1 Introduction

The Visualization layer of the eScop architecture makes possible the visualization of the Manufacturing system on web-enabled devices. The visualization is provided by a Human Machine Interface (HMI) which is a User Interface (UI) to facilitate interaction between a human and a manufacturing or process control system. It help production monitoring systems to collect information at run time to enable production team to respond in a timely manner to handle production related issues that may arise and also assists to produce products with available resources at the same time improving quality matters and reducing overheads (Subramaniam et al., 2007).

The HMI is composed of simple or complex visual symbols. The complex symbols are those with higher levels of details and usually combination of simple symbols. The simple symbols are defined by Scalable Vector Graphics (SVG). SVG is a XML-based vector image format for graphics.The SVG specification is an open standard recommended by World Wide Web Consortium (W3C) and all modern web browsers like Mozilla Firefox, Internet Explorer, Google Chrome, etc., have some degree of SVG rendering support. The SVG images can be created with different shapes and customizable features (W3C, 2011).

The set of features required to compose visualization of SVG symbols are modelled as SVG ontology, which was extended from SVG graphics ontology (Mathis, 2013). It provides an ontology format for visualization representation. Storing all system information in same format facilities easier system management and supports possible extensions in the system as the information is represented in universal format and similar tools can be used for different aspects of the system.

The remaining part of this chapter is structured as follows: The next section describes the concept of visualization for eScop architecture. The third section explains how the graphics are linked to actual component or real time data of system, while fourth and fifth sections address representation of graphics itself within ontology.

2 Visualization for manufacturing systems

Visualization aims to present information about manufacturing system in a clear and effective way using graphical representation. The VIS requests the Representation layer (RPL) for the visualization information and creates different displays or user views. The RPL includes eScop MSO which is a knowledge base storing all the information about manufacturing system. The SOA based systems offer the capability of self-description of devices on factory shop floor. They lead to broader availability of data (Vallipour et al., 2009) and as a result we can obtain any information about the system without need for custom integration or reconfiguration of the visualization. This makes easier to access data and also to store it and update in the ontology. This information can be reused to support visualization since the graphics represents and exhibits behavior of devices or components on shop floor. Such information that can be used to generate the visualization may refer for example to: what are the devices on shop floor, how each component is connected to other, which process is performed etc. VIS retrieves from MSO the visualization information needed for each display.

After the VIS receives the necessary information from RPL, visualization display is created. The display includes multiple simple visual elements or complex elements which depicts the behavior of actual components of the system. Some examples of visual elements include screen, table, graph, map, button etc.

The ontology to represent the visualization knowledge must include the information on composition of the display. It must define the mapping between the visual elements and real components or data of the system. It must also represent the graphics meaning the definition of its shape, size, position, color and other features. The visualization representation in MSO is presented in the next section.

3 Manufacturing System Ontology (MSO) for Visualization

The MSO represents the knowledge on different faces of the Manufacturing system. The ontology to support visualization must be able to represent all the information necessary for visualization of the system. As discussed before, the VIS creates displays using visual elements that could be connected to data. The link between visual element and actual component of the system is stored in MSO (Fig. 1).

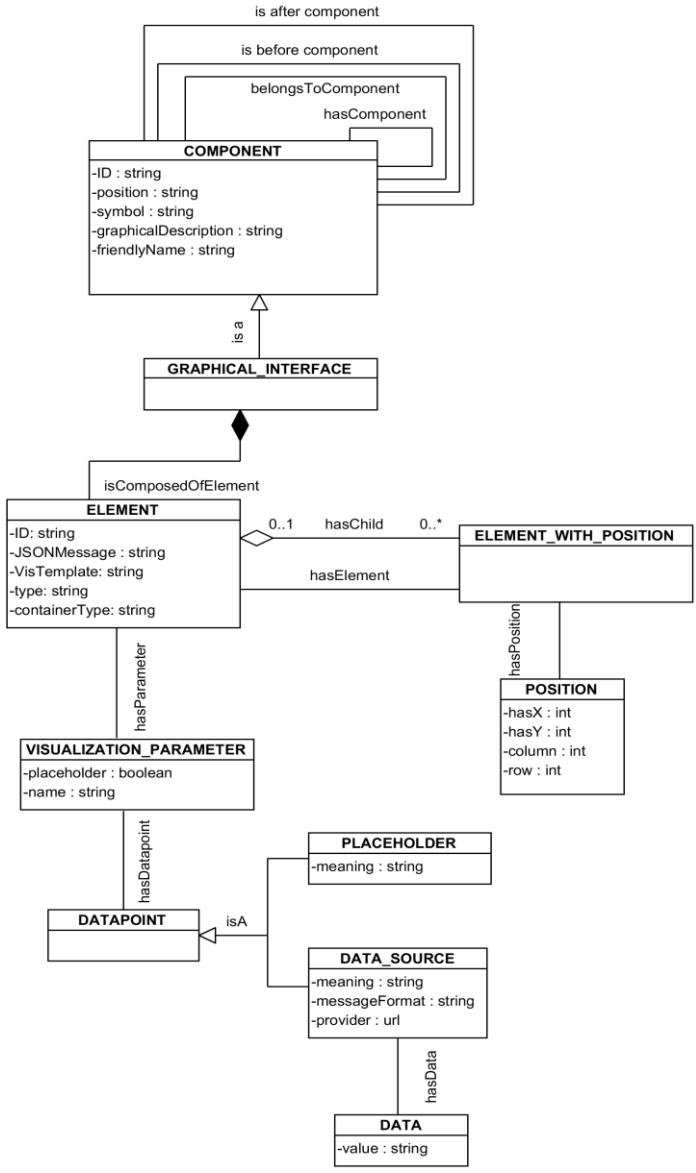


Fig. 1. Visualization representation in MSO

Graphical interface is a component of system. All other components like transporter, sensor etc., can have graphical interface that links them to the visual element. Hence in the MSO, the GRAPHICAL_INTERFACE class is a sub-class of the COMPONENT class and it is linked to the ELEMENT class by the property isComposedOfElement.

The ELEMENT class represents the visual elements like screens for operator display, shop floor layout, table and so on. Elements can have child elements at defined positions. For example an operator screen element can have a table element at some position on the screen. The ELEMENT class is linked to the ELEMENT_WITH_POSITION class using the hasChild property. The ELEMENT_WITH_POSITION class is linked to the ELEMENT class and the POSITION class using the hasElement and the hasPosition properties.

The positions are defined based on container type. For example an element with container type canvas has its child element at position defined by x,y coordinates. But if the container type is table then the child position is defined by rows and columns. The containerType property is used to know the container type so that positions can be defined based on it.

Element has some visualization parameters like sensor values, temperature etc., which is represented with VISUALIZATION_PARAMETER class. And it has some datapoint which includes placeholder and data source. When placeholder is false, it means data is available from data source and reverse for placeholder true. The DATASOURCE class defines the meaning of data, message format in which data will be available from source and the provider URL. Data class defines the data.

When VIS requests RPL for visualization information of an element, the information represented in the ontology will be retrieved and sent to VIS. The retrieved information must be sent in some compatible format to be processed by VIS. In this chapter JSON format is considered. The JSONMessage datatype property links each element to a JSON message which describes the visualization information of that element. The message is formed by using the knowledge in ontology about that element like what are its child elements, their position, etc.

4 SVG Ontology

The Scalable Vector Graphics is used to define vector-based graphics for the web. It defines the graphics in XML format. SVG is a World Wide Web Consortium recommendation and supported by major modern browsers. The simple visualization symbols that are used to make the HMIs are SVG symbols. The SVG symbols have different shapes that are used to make visualization. SVG ontology is a knowledge base for representing the SVG format. A basic SVG symbol defines the shape, position of shape (X, Y), size (height, width) and style (fill, stroke and color). These features are modelled in the SVG ontology.

The terminology used for modelling is based on W3C standard for SVG. The tool used for modelling is Olingvo: a graphical application for creating and editing OWL models. The main classes are Shape, Style, Fill, Stroke and Color. The class hierarchy of SVG ontology is shown in Fig. 2. The properties defined in the SVG ontology are given in Fig. 3.

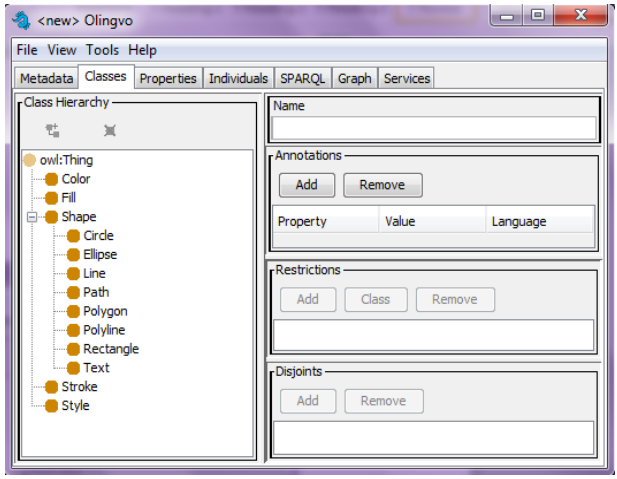


Fig. 2. Class hierarchy for SVG ontology

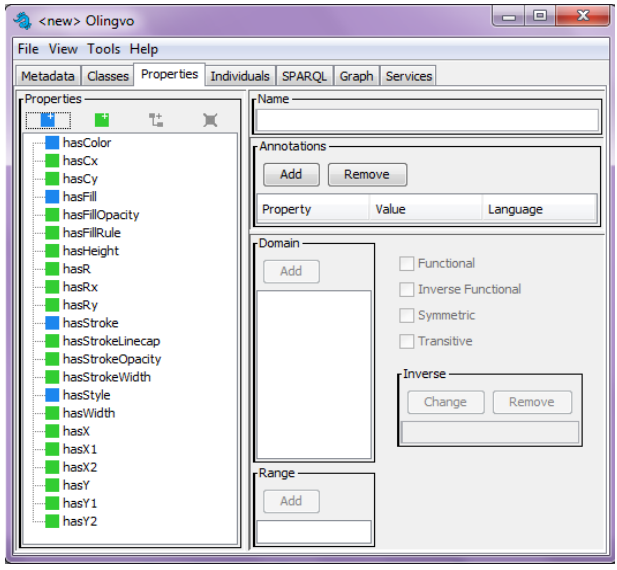


Fig. 3. Properties for SVG ontology

The Shape class is defined to contain different shapes like circle, line, rectangle etc. The location of the shape and size is defined by the datatype properties hasX, hasY, hasHeight and hasWidth. The style for the shape is defined with object property hasStyle.

The different styles for the shapes are given by the Style class. The Shape class links the Style class using object property hasStyle. The style includes color, filled/not fillet, stroke and stroke color. To include the style concept in ontology classes like Color, Fill and Stroke are defined. The Style class is linked to Fill Class and Stroke

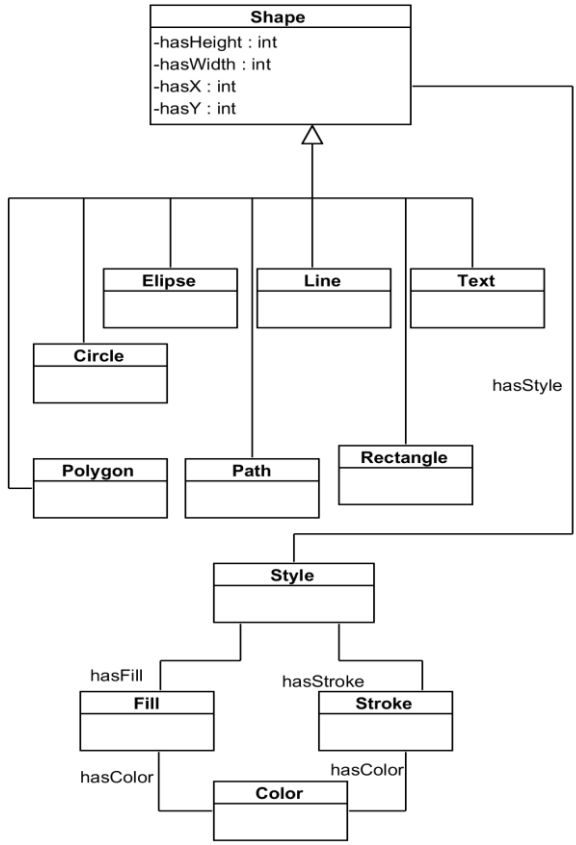


Fig. 4. Class diagram for SVG ontology

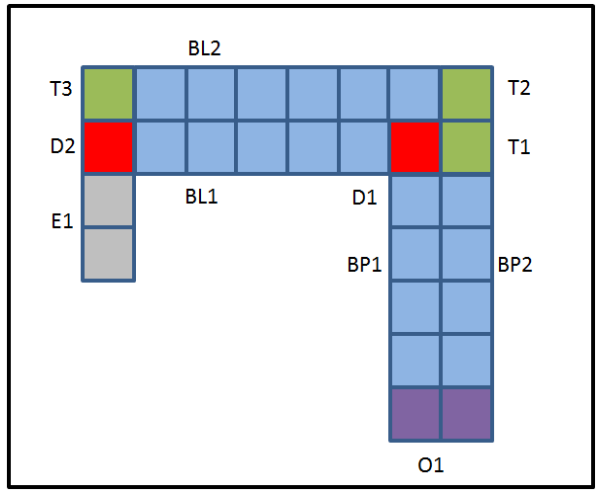


Fig. 5. Visualization graph using SVG ontology

class using object properties hasFill and hasStroke. The Fill class defines color using the object property hasColor. The Fill Class also defines no-fill condition. The Stroke class is also like the Fill class and defines color for stroke using object property has-Color. The Color class has all the possible colors as instances. The class diagram for SVG ontology is presented in Fig.4.

The Fill class defines color using the object property hasColor. The Fill Class also defines no-fill condition. The Stroke class is also like Fill class and defines color for stroke using object property hasColor. The Color class has all the possible colors as instances. The class diagram for SVG ontology is presented in Fig.5.

The SVG format is represented in the knowledge base in this manner. By integrating the SVG ontology with MSO, the simple symbols required for visualization could be stored in ontology. This information is used by VIS agent to make the HMIs.

A simple example, visualization of factory shop floor layout of logistics system using the SVG ontology, is discusses below. The shop floor layout has conveyors: sequence buffers (BL1, BL2, BP1 and BP2), decision module (D1 and D2), output module (E1), picking station (O1) and transfer module (T1, T2 and T3). The visualization graph obtained from using the SVG ontology along with MSO for the logistics system example is shown in Fig.5. Using the SVG ontology, the conveyor shapes, size, height, width, position (x, y), fill, stroke etc. are stored and are retrieved by VIS to make visualization screen.

5 Symbol library for complex symbols

The visualization screen apart from simple symbols can be composed of complex symbols. Complex symbols can also be combination of simple symbols but with high level of details in the graphics. This makes difficult to represent symbols in ontology since it might lead to overload of data and to define such complex symbols, and complicates the process of information retrieving and composing the symbol. To deal with these issues, the concept of symbol library can be used. Symbol libraries are used to store, external to MSO, the ready-made templates for complex symbols. The complex symbols can be generated using some configuration tool and stored in symbol library.

When a complex symbol has to be defined for representing a component, then the component can be mapped to the corresponding template in symbol library. The template is also reused when many components have to be represented by same type of complex symbol. Within the eScop complex symbols are created with ICONICS GraphWorX64 (ICONICS, 2013), which is a rich HMI and SCADA data visualization tool for building scalable vector-based graphics.

VIS uses the information from ontology about visualization along with the symbol library to generate visualization display. The information sent to VIS from RPL includes information on screen composition and metadata defining visualization specific information e.g. position of visual elements in the screen. It also provides the mapping to the visual symbol for each element. The mapping can be to symbols in symbol library or simple SVG symbols represented directly in MSO. With the visualization information obtained from RPL, the VIS composes the screen.

References

ICONICS (2013). GraphWorX64. http://www.iconics.com/Home/Products/HMI-SCADA/GENESIS64/GraphWorX64.aspx. Accessed 2014.XI.16.

Mathis, R.M. (2013). SVG Graphics Ontology. http://www.mathiswebs.com/ontology/. Accessed 2014.XI.16.

Subramaniam, S. K. A., Husin, S. H. B., Yusop, Y. B., & Hamidon, A. H. B. (2007). Real time production performance monitoring system a production aid all industries. Proceedings of the 6th *WSEAS International Conference on CIRCUITS, for SYSTEMS, ELECTRONICS, CONTROL & SIGNAL PROCESSING, Cairo, Egypt, 2007.12.29-31*: 181-184.

Valipour, M. H., Amir Zafari, B., Maleki, K. N., & Daneshpour, N. (2009). A Brief Survey of Software Architecture Concepts and Service Oriented Architecture. *Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology (ICCSIT'09)*: 34-38.

W3C (2011). Scalable Vector Graphics. World Wide Web Consortium. http://www.w3.org/TR/SVG/. Accessed 2014.XI.16.

PART THREE

TOWARDS (WEB-)SERVICE ORIENTED

MES ARCHITECTURES

State-of-Art of Manufacturing Execution Systems

A Technology Review

Xiangbin Xu[1], Sari Räsänen[2], Roberto S. Camp[3]

[1,2]Tampere University of Technology, Tampere, Finland
[3]FluidHouse Oy, Jyväskylä, Finland

[1,2]{xiangbin.xu, sari.rasanen}@tut.fi, [3]roberto.camp@fluidhouse.fi

Abstract. Over the past twenty years industrial manufacturers around the world have actively recognized the significance of Manufacturing Execution Systems (MES) with respect to the optimization of manufacturing processes. Hierarchically between the enterprise management level and the shop-floor automation level, MES aims to collect and convey real-time information of production activities such as material flow, equipment state and labor deployment, to help management make fully informed decisions. On the other hand, MES emphasizes the optimization of manufacturing processes from order launch to finished goods by taking account of all manufacturing related elements. Hence this automated information flow between the company level and field level boosts the productivity and efficiency of operational assets as well as on-time delivery, inventory turns. This chapter introduces the evolution, concept, fundamental functions of MES and actual implementation in the industry from a technology point of view and discusses the limitations and trends of MES in terms of solving new challenges in front of industrial manufacturers. Eventually, the state-of-art of Open Knowledge Driven Manufacturing Execution System is presented.

1 Introduction

The concept, functions and current status of manufacturing execution system (MES) have to be understood before developing more advanced and feasible features. The Manufacturing Execution Solutions Association (MESA) defines that MES are systems that deliver information enabling the optimization of production activities from order launch to finished goods (MESA, 2000). Using current and accurate real-time data MES system guides, responds to, and reports on plant activities when they occur. The resulting rapid response to changing conditions, coupled with a focus on reducing non-value added activities, drives effective plant operation and processes. From the technological point of view, while MES vendors may provide divergent MES solutions based on different architectures, in general MES is supposed to provide elementary functions such as resource allocation and status, operations/detail scheduling and dispatching production units. These functions as a whole enable MES to fully manage and optimize manufacturing activities, although MES solution providers may also deliver their specific functions. Therefore, a concise analysis of the MES functions

according to the production type is pivotal for the success and usability of MES solutions. Additionally, with increasing economic globalization, manufactures face new challenges and opportunities. Case in point, the need for more diverse, customized products demand more responsive and flexible reconfiguration of manufacturing processes, shorten product life cycles that require less production time, the generality of the situation being that the merging of factories necessitates the integration of manufacturing activities. All of these factors contribute to the need of more flexible, more intelligent MES systems in order to maintain the competitiveness of manufacturers. Fortunately, the advances of technologies make it possible for MES to fulfill these needs. The increasing computing power of embedded devices makes equipment capable of performing more complex functions. Big data technology facilitates collecting varying unstructured data from shop-floor. Cloud computing and the Internet of things makes systems more available everywhere. These technological improvements have significant potential to generate more scalable, flexible MES solutions for the future.

2 Evolution of MES

Nowadays, information systems have become ubiquitous in various activities of manufacturing, ranging from enterprise management to the shop floor control. Their usage significantly boosts the productivity and efficiency of industrial production. Back in the 1970s, the use of computers in production activities was limited to two types of systems. One type was the mainframe-based electronic data processing system, used in production management and materials procurement. The other type of system was the microcomputer/minicomputer-based factory automation systems used with robots and automatic manufacturing equipment (Tsuratani, 1990). For more than 30 years, companies have invested in these two types of information systems to achieve productivity gains such as the Enterprise Resources Planning (ERP) system which is focused on planning. ERP is an industry term for integrated, multi-module application software packages that are designed to provide information systems for multiple business functions as order entry, general ledger, purchasing, warehousing, transportation, human resources and manufacturing (Peniak, 2006). Meanwhile, the developments of PLC, DCS, SCADA and fieldbus systems facilitate the shop floor control. The gap between these two types of information systems requires a middle layer; once the plan set by ERP has been developed, there must be a translation of the plan that deals with real resources that are currently available. To bridge this gap, what is necessary is a method to take input from the planning system and translate that plan into a language that fits the plant floor and the resources required to execute the plan—a major role for MES (McClellan, 2001). This kind of vertical integration is realized by MES as illustrated in Fig. 1. The three layers can also be referred to as Company Management (for which ERP systems are the most common tools), Production Management (done by MES), and Production (supported by systems for machine control and acquisition of manufacturing data) (Schmidt et al., 2009).

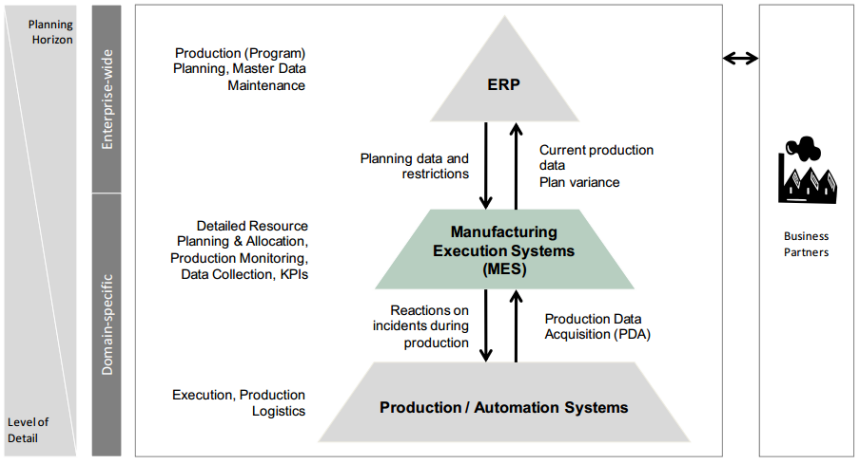


Fig. 1. MES as connector between ERP and shop floor (Schmidt et al., 2009)

In principle, MES may interact with many other information-intensive systems as shown in Fig. 2. MES systems have a much broader definition than what they are actually used for. A lot of its originally intended functions have been deferred to the ERP level solving some problems but causing others. In some cases DCS systems can include MES functions and are therefore sometimes incorrectly considered MES systems of the Process Industry. MES in the process industry are placed in between the DCS and the ERP level bridging the existing gap. These also manage recipe systems and batch scheduling. Overall MES systems in discrete, batch and process manufacturing serve the exact same purpose - bridging the gap between the shop floor and the enterprise level. In some cases this implies linking the ERP to the PLC control system, in others it implies linking to the SCADA and DCS systems. The main difference resides in MES functions implemented and the focus given.

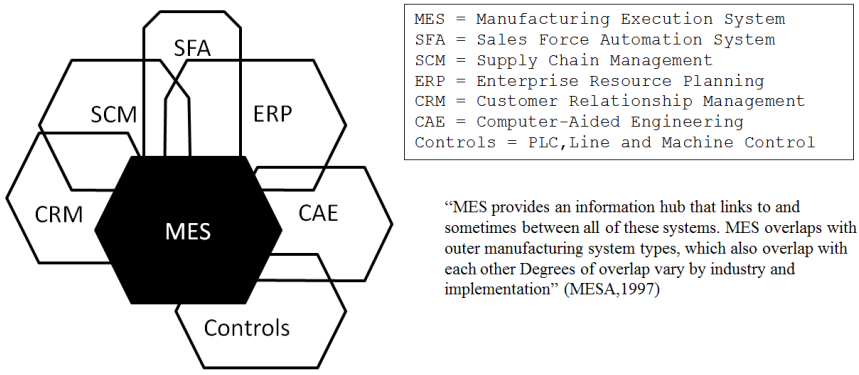


Fig. 2. An enterprise information context model (adapted from Qiu and Zhou, 2004)

3 MES Functions

A clear understanding of the functionalities offered by a complete MES is helpful for identification of its numerous benefits and also for the development of MES. In 1997, MESA defined the 11 functions of MES in its MESA-11 model (Leibert et al., 1997; Fraser et al, 1997). Other MES related white papers, such as (Barkmeyer et al. 1999), present MES functions that are slightly different than MESA-11 model. However, the MES functions can be generalized by referring to the MESA-11 model as follows:

- Resource Management: Ensures availability of resources including machines, tools, materials and other equipment for processing. Manages and processes work-in-process (WIP) stocks. It maintains detailed history of resources. Alter predefined schedule on the factory floor. Manages reservation and dispatching to meet operation-scheduling objectives.
- Operations/Detail Scheduling: Checks production restrictions including evaluation of available resources and conditions. Conducts detailed scheduling: plan time and equipment loading, adjusts to shift patterns, optimize production plans and sequences. Implements sequence scheduling. Convert production requirements to production orders. Disposes production orders.
- Manufacturing Control: Monitors production. Manages order sequence. Compares planned and actual production situation. Actualizes production plans in real-time. Supports the decision of operators. Records wastes. Informs inventory control of material movements. Transmits production data to production documentation.
- Production documentation: Records and reports actual production results in an up-to-the-minute time frame. Long-term manufacturing analysis report. Provides production evaluation report and visualizes report data.
- Data Collection/Acquisition: Pre-processes, verifies and aggregates data. Displays production data. Collects data from factory floor either manually or automatically in an up-to-the-minute time frame.
- Labor Management: Provides status of personnel in and up-to-the-minute time frame. It includes time and attendance reporting, certification tracking, as well as the ability to track indirect activities such as material preparation or tool room work as a basis for activity based costing. It may interact with resource allocation to determine optimal assignments.
- Quality Management: Compiles quality planning. Evaluates testing analysis results. Provides suggestions of actions to correct problems and measures. Determines and manage rework. Manages measuring and testing equipment. Transmits quality data for production documentation.
- Maintenance Management: Tracks and directs the activities to maintain the equipment and tools to insure their availability for manufacturing and insure scheduling for periodic or preventive maintenance. Triggers alarms to immediate problems and informs maintenance manager. Collects historical data of equipment events or problems.

- Processes Management: Manages the sequence of operations within a manufacturing process. Provides process specification or product specifications. Balances load and routing of WIP in real-time. Manages exceptions or deviations.
- Product Tracking and Genealogy: Provides the visibility to the status of work at all times and its disposition. Provides on-line tracking of information about components assembled into parents. Information may include who is working on it; components materials by supplier, lot, serial number, current production conditions, and any alarms, rework, or other exceptions related to the product. This record allows traceability of components and usage of each end product. Stores production/product history data. WIP tracking.
- Performance Analysis: Provides up-to-the-minute reporting of actual production operations results along with the comparison to past history and expected business result. Performance results include such measurements as resource utilization, resource availability, product unit cycle time, conformance to schedule and performance to standards. Draws on information gathered from different functions that measure operating parameters. Triggers alarms when parameters deviate from ranges. These results may be prepared as a report or presented on-line as current evaluation of performance. Although it is part of MES system, most of the conventional systems delegate the analysis to the ERP level. Hence it makes a MES system a data collector for ERP. As a result the flexibility of production suffers, because of the rather slow reaction of ERP systems (Bratukhin and Sauter, 2010).

These eleven functions as a whole describe all the potential functions that a MES system should provide at the time. In 2003, MESA extended the MESA-11 into the collaborative MES model shown in Fig. 3. MES functions described above were redesigned and merged to eight major functions. In the collaborative MES model, detailed scheduling, document control and maintenance management functions may be separate or part of MES.

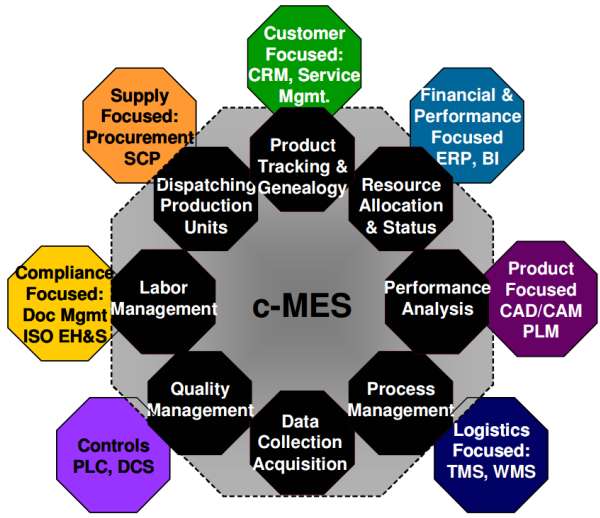


Fig. 3. MESA Collaborative MES (cMES) model of eight major MES functions (MESA, 2004)

4 Available MES solutions and deficiencies

MES products are provided by different types of solution providers. They can be classified to (Table 1):

- Automation vendors such as: ABB, GE Intelligent, Honeywell, Invensys Operations Management, Rockwell Automation, Siemens,
- ERP venders, e.g.: SAP and Oracle,
- Pure-play MES solution providers, e.g.: Camstar System, Werum Software and Systems, Aprico Corporation,
- Niche solution providers for quality management and manufacturing intelligence, e.g.: Pilgrim Software, Sparta Systems, OSIsoft.

In addition to these commercial MES productions, there are also open-source solutions available such as an online production management application offered by Qcadoo and FACTORITY by Assa Systems (ASSA Systems, 2009). MES product from each provider may have different architectures, platforms and variable functionalities.

Although the current MES solutions intend to cover all necessary aspects of manufacturing processes, some major issues still exist. First, integration of information system throughout the enterprise is a demanding task particularly with systems from different vendors sometimes even impossible. Companies such as Siemens or SAP

Table 1. Features of the leading MES solutions

MES product	Wonderware (Wonderware, 2012)	SIMATIC IT (Giacomo, 2012)	ABB MES (ABB, 2010)	Factory Talk Production Centre (Rockwell Automation, 2012)
Vendor	Invensys Operations Management	Siemens	ABB	Rockwell
Standard MES functions	All	All	All	Maintenance management not included
Other functionalities	Statistical Process Control	Reporting, Inventory Operation, SAP integration	Warehouse management, Assess management	Supplier Management, Production Inventory, Analysis and reporting tool (identification and root cause analysis of KPI), Materials Management
Other information	Support ISA-95 Integration with SAP, Microsoft Dynamics, Oracle, Infor et al. Uses ArchestraA technology	Conforms ISA-95 Interaction with product specification management, ERP, other business systems	ISA-95 based solution Integration with ERP	

have attempted to solve this issue by interfacing MES products and ERP systems. However this type of interface only works for limited vendors. Collaboration between various MES providers and open source software seeking to provide communication between differing systems has begun to solve this problem.

The deficiency of re-configurability is another weakness of MES as they are unable to support flexible manufacturing. The contemporary MES usually uses a relatively rigid structure and model. It is reasonable because most manufacturing enterprises have a certain type of configuration. However industrial manufacturers have begun to realize the need of flexible manufacturing in order to quickly react to market demand and product customization. In this case most MES are not capable of easily adapting to this reconfiguration in production process and associated data collection. It is true that the current MES solutions can be configured for various manufacturing environment, but they lack the speed of adaptation and reconfiguration after deployment.

System intelligence is another issue that MES can improve. MES provides functions of data collection and reporting for the production manager to have a real time view of the plant but few of them have deeper analysis on the data collected. The detailed analysis on the production related data has potential benefit for decision makers. Moreover, fewer MES solutions have the intelligence to make decisions automatically based on the analysis of data.

5 Trends of MES

Emerging technologies in the ICT domain pave the way for further improvements of MES systems. In this section several potential technologies are examined to explore how they can benefit the development of MES in the future.

5.1 Cloud computing

Cloud computing has been attracting consistent attention in the computer world. It refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services. The services themselves have long been referred to as Software as a Service (SaaS). The datacenter hardware and software is what can be called a Cloud (Armbrust et al., 2009).

From the perspective of the end users, Cloud computing delivers computing services over the Internet. Cloud services allow individuals and businesses to use software and hardware that are managed by third parties at remote locations. The most obvious advantage of Cloud computing is resource saving - both equipment resources and maintenance resources; it reduces the cost and complexity of owning and operating computers and networks since cloud users do not have to invest in information technology infrastructure, purchase hardware, or buy software licenses. It also reduces the cost and time used to maintain the local applications for instance installing new packs, backup and patching. Merrill Lynch projected the cost advantages of Cloud Computing to be three to five times for business applications and more than five times for consumer applications (Merrill Lynch, 2008).

Some Cloud based solutions are already available for PLM features, for Engineering business process function, CAX integration (Parihar, 2012). From technology point of view, MES vendors could provide MES services in Cloud. So that manufacturer does not need to spend extra capital on purchasing IT infrastructures, instead only the access to the Internet. In this case it requires the production machine should also have the capability of communicating with Cloud in order to collect data. One solution is the introduction of smart RTUs that can handle this requirement. This type of RTU has been available and used in the industrial world (Minor et al., 2012). In addition, Could-based MES solutions have to overcome the common downsides of Cloud computing. The first issue is Internet outrage or problems. It means that the users cannot access the applications on the Cloud. Second, anything in the Cloud is not private. The users do not know where the data is, where the application is executing. To this end, private Clouds are gaining popularity. In 2012 the BMW group already started to implement private cloud architecture. The private Cloud strategy implemented at BMW aims to a target of zero downtime, provision of self-service, automated, rapid and elastic provisioning and release of services and user choice (Alliance, 2012).

5.2 Big Data

Big Data is a loosely defined term used to describe data sets so large and complex that they become awkward to work with using standard statistical software (Snijders et al., 2012). In fact, data have become a torrent flowing into every area of the global economy (Economist, 2010). Some studies have tried to estimate the total amount of data generated, stored, and consumed in the world (Short et al., 2011; Hilbert and López, 2011). They all point to exponential data growth in the years ahead.

Industrial manufacturing environment is one of the very places where large and heterogeneous data are generated every day. The data in the manufacturing environment come from various sources; terabytes of sensor readings and equipment states can be generated during operations, this data is often under-utilized. Factory engineers use the data usually for historical evaluations, equipment failure analysis although it could be used for predictive analysis. Quality data are collected when manufactures inspect incoming material, the semi-finished and finished products. The share of this data and analysis could benefit other production lines. Data could also be received when manufacturers receive raw material or components from the suppliers. This information gets stored and is valuable for analysis but not always easy for quality analysts and engineers to access. So with respect to MES, how can big data technologies create value? MES could make data more transparent so that it can be integrated with those from R&D and engineering, to enable concurrent engineering which cuts time to market and improves quality.

Using data to analyze variability in production performance, which either occurs naturally or is generated by controlled experiments, and to understand its root causes, can enable production managers to manage performance to higher levels. MES could provide sophisticated analytics that can substantially improve decision making, minimize risks, and unearth valuable insights that would otherwise remain hidden. In

some cases, decisions will not necessarily be automated but augmented by analyzing huge, entire datasets using big data technologies rather than just smaller samples that individuals with spreadsheets can handle and understand. For instance early-warning analytics can enable operators to proactively address potential quality and performance issues before they become customer problems. The increasing deployment of the "Internet of Things" is also allowing MES to use real-time data from sensors to track parts, monitor machinery, and guide actual operations.

5.3 Knowledge-based systems

A knowledge-based system (KBS) is a computer program that reasons and uses a knowledge base to solve complex problems. KBS are based on a coded body of human knowledge represented through a mathematical model. The task of developing a KBS is generally known as knowledge engineering, while the specific task of collecting human knowledge and representing it through a mathematical model is known as elicitation (Piatti et al., 2010).

The knowledge base that represents facts about the world is often in some form of ontology. With the availability of advanced computing facilities, more demanding tasks that might require intelligence are drawing attention. The advantages of a KBS are that it can act as an expert on demand anytime and anywhere. It can save money by allowing users to function at a higher level and promoting consistency. It is a productive tool that offers collective knowledge of one or more experts. Akerkar and Saja (2010) summarized the differences between a traditional computer system and a knowledge based system, as shown in Table 2.

Table 2. Comparison of Knowledge-based and Computer-based information system

Traditional Computer-Based Information System	Knowledge-Based System
Gives a guaranteed solution and concentrates on efficiency	Adds power to the solution and concentrates on effectiveness without any guarantee of solution
Data and/or information processing approach	Knowledge and/or decision processing approach
Assists in activities related to decision making and routine transactions; supports need for information	Transfer of expertise; takes a decision based on knowledge, explains it, and upgrades it if required
Examples are TPS, MIS, DSS, etc.	Examples are expert systems, CASE-based system, etc.
Manipulation method is numeric and algorithmic	Manipulation method is primarily symbolic/connectionist and non-algorithmic
These systems do not make mistakes	These systems learn by mistakes
Need complete information and/or data	Partial and uncertain information, data, or knowledge will do
Works for complex, integrated, and wide areas in a reactive manner	Works for narrow domains in a reactive and proactive manner

For MES as supervising and controlling systems it is evidently required to collect and use data and information from different heterogeneous sources. In an ideal environment, a homogeneous data format for the exchange of information is used. This format has to be as generic as possible, but as detailed as required. MES therefore require two specific paths in information management to increase efficiency. This is why a knowledge-based system could benefit MES in two ways: On one hand, an Ontology knowledge-base is able to process and fuse data from heterogeneous sources. On the other hand, the Ontology knowledge-base enables knowledge sharing and reusability, which significantly simplifies the development process of MES.

6 Conclusions

MES is becoming a commonly used technology which aims to bridge the gap between ERP system and shop floor control system, providing a real time overview of the plant and executing the production plan. Several organization have managed to create models and standard to help implement MES. The models and standard can be generalized into eleven basic functions. These functions intend to cover all the aspects of production activities. Although there are a wide range of MES solutions available on the market, deficiencies exist and further development of MES could take them into consideration. With the demand of fast reconfiguration of plant to meet market changes and customization, a more open, interoperable and intelligent MES could further improve the productivity of manufacturing process. In this sense, emerging ICT technologies such as Cloud computing, Big data and Knowledge-based systems have significant potential to improve the development of next generation MES.

References

ABB (2010). Collaborative Production Management: A new level of operational excellence with ABB's Manufacturing Execution Solution. ABB.

Akerkar, R., & Sajja, P. (2010). Knowledge-Based Systems. Jones & Bartlett Publishers, p. 18.

Alliance (2012). Open Data Center Alliance: The Private Cloud Strategy at BMW. Alliance.

Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., & Zaharia, M. (2009). Above the clouds: A Berkeley view of cloud computing. *Technical Report No. UCB/EECS-2009-28*, University of California, Berkeley.

ASSA Systems (2009). http://www.factority.org/index.html. Accessed 2014.11.20.

Barkmeyer, E., Denno, P., Feng, S., Jones, A., & Wallace, E. K. (1999). NIST Response to MES Request for Information. National Institute of Standards (NIST), Gaithersburg, 1999.

Bratukhin, A., &Sauter, T. (2010). Bridging the Gap between Centralized and Distributed Manufacturing Execution Planning. *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation (ETFA'2010)*: 1-8.

Economist (2010). A special report on managing information: Data, data everywhere. February 25th, 2010.

Fraser, J. et al (1997). MES Explained: A High Level Vision for Executives. *MESA White Paper*, No. 6.

Hilbert, M., & López, P. (2011). The world's technological capacity to store, communicate, and compute information. *Science*, Vol. 332(6025): 60–65.

Leibert, J., Muroski, M., Fraser, J., Davis, T.L., Towle, S., Tebbenhoff, P., Navarro-Robertroy, S., Huff, D., Chance, C., Wells, M., Woods, C., Kukla, D., Hakanson, B., & Zailyk, S. (1997). MES Functionalities & MRP to MES Data Flow Possibilities. *MESA White Paper*, No. 2.

McClellan, M. (2001). Introduction to manufacturing execution systems. Paper presented at MES conference & exposition, Baltimore, Maryland, 4–6 June 2001.

Merrill Lynch (2008). The Cloud Wars: $100+ billion at stake. Merill Lynch, May 7, 2008.

MESA (2000). Controls Definition & MES to Controls Data Flow Possibilities. *MESA White Paper*, No. 3.

MESA (2004). MESA's Next Generation Collaborative MES Model. *MESA White Paper*, No.8.

Minor, J., Garcia Izaguirre, J., Martinez, J., Lobov, A., & Martinez Lastra, J.L. (2012). Evaluating Service-Oriented Orchestration Schemes for Controlling Pallet Flow. *Proceedings of the Seventh International Conference on Systems (ICONS'2012)*: 88-92.

Parihar, A.S. (2012). PLM and Cloud computing. http://www.infosys.com/engineering-services/features-opinions/Documents/social-plm.pdf. Accessed 2014.11.20.

Peniak, P. (2006). Implementation of information systems in manufacturing area. *Advances in Electrical and Electronic Engineering*, 5/3:120-123.

Piatti, A., Antonucci, A., & Zaffalon, M. (2010). Building knowledge-based systems by credal networks: a tutorial. In A.R. Baswell (Ed.), *Advances in Mathematics Research*, Vol. 11, Nova Science Publishers.

Qiu, R,G., & Zhou, M. (2004). Mighty MESs: State-of-the-Art and Future Manufacturing Execution Systems. *IEEE Robotics & Automation Magazine* 11/1: 19-25.

Rockwell Automation (2012). Automotive MES Solutions. Improve your time to market and reduce costs. Rockwell Brochure.

Schmidt, A., Otto, B., & Kussmaul, A. (2009). Integrated Manufacturing Execution – Functional Architecture, Costs and Benefits. Institute of Information Management at University of St. Gallen.

Short, J.E., Bohn, R.E., & Baru, C. (2011). How Much Information? 2010 Report on Enterprise Server Information. http://hmi.ucsd.edu/pdf/HMI_2010_EnterpriseReport_Jan_2011.pdf. Accessed 2014.11.20.

Snijders, C., Matzat, U., & Reips, U.-D. (2012). 'Big Data': Big gaps of knowledge in the field of Internet. *International Journal of Internet Science*, 7/1: 1–5.

Tsuratani, T. (1990). Manufacturing information systems and productivity improvement activities. *Proceedings of the 9th IEEE/CHMT International Electronic Manufacturing Technology Symposium, Competitive Manufacturing for the Next Decade (IEMT'1990)*: 12 – 16.

Torre, G. (2012). Browsing through Success: A panorama of MES Applications with SIMATIC IT. http://www.docstoc.com/docs/159681874/Browsing-through-Success-Siemens. Accessed 2014.11.20.

Wonderware (2012). MES software 2012. The Foundation for Operational Excellence. Datasheet. http://global.wonderware.com/EN/PDF%20Library/Datasheet_Wonderware_MES2012. Overview_11-12.pdf. Accessed 2014.XI.20.

Developing Open Knowledge-Driven Manufacturing Execution System

Sergii Iarovyi and Xiangbin Xu

Tampere University of Technology, Tampere, Finland

{sergii.iarovyi, xiangbin.xu}@tut.fi

Abstract. Currently Manufacturing Execution Systems (MES) are being important part of factory infrastructure. Such systems enable improvement of factory performance in terms of efficiency, productivity, etc. Due to complex functionality of MES they tend to be offered as complex proprietary software. Integration of them in a factory ecosystem is difficult task. This chapter suggests and discusses an open knowledge driven (OKD) approach for MES, to provide an extendable and smart information system. The OKD approach reduces costs of introduction and reconfiguration of MES systems in modern factories.

1 Introduction

McClellan (1997) proposed a following definition for MES systems: *"A manufacturing execution system is an online integrated computerized system that is the accumulation of methods and tools to accomplish production"*. The term online in this definition is mainly related to connection of MES on one side to factory shop floor, and on other side to Enterprise Resource Planning (ERP) system, linking the planning level with operation of manufacturing equipment at shop floor level.

The author emphasizes word integrated, as functionality available in such integrated system goes beyond functionality of its isolated parts. More over Kletti (2007) states vertical integration of MES with other levels of company is as important as horizontal integration of parts of MES level functions (Fig. 1). This relates to position of MES within the automation pyramid.

Application of MES may significantly improve performance of manufacturing systems. According to a study performed by Manufacturing Enterprise Solutions Association (MESA) and the Industry Directions Inc. analyst firm a clear correlation between the use of MES and enterprise performance improvements exists. Important Key Performance Indicators (KPI) in companies using MES are significantly more likely to be improved comparing to the ones not using such systems (Accenture, 2010).

Functionality of MES has been defined by several organizations. Among the most influential there are those by Manufacturing Enterprise Solutions Association (MESA, 2014) and by Verein Deutscher Ingenieure (VDI) (VDI, 2014). The functionality of MES is similarly grouped by both organizations (Kletti, 2007).

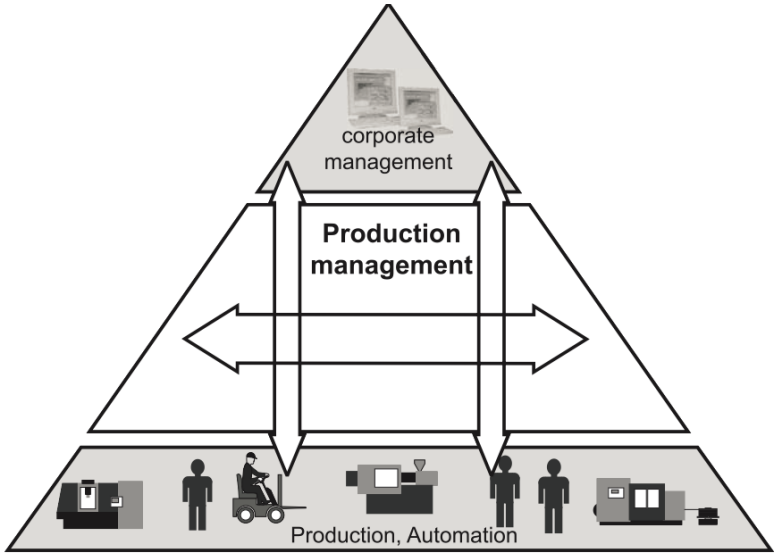


Fig. 1. Vertical and horizontal integration of MES (Kletti, 2007)

Table 1. MES functionalities defined by MESA and VDI

MESA	VDI
• Operations / detail scheduling	• Detailed planning and detailed scheduling control
• Resource allocation and status	• Operating resources management
• Document control	• Material management
• Dispatching production units	• Personnel management
• Performance analysis	• Data acquisition and processing
• Labor management	• Interface management
• Maintenance management	• Performance analysis
• Process management	• Quality management
• Quality management	• Information management
• Data collection and acquisition	
• Product tracking and genealogy	

Industry demands not only for the named functions, but also for a manner of their implementation. R. Martin from AMR Research has set in the end of 00's two requirements for contemporary MES: being web-based services and having more flexible data model. Consumers desire to select the best-of-breed product, forces contemporary MES software vendors to enable integration possibilities to interconnect with shop floors, ERP systems and other elements of contemporary factory. Discussing future of MES Johnson (2010) suggests that the trend will drive modularization and separation of MES Core functions from other functionalities, leading to growth of a market for modules. This follows the concept of vertical and horizontal integration, providing more freedom to users, to select tools to be used. Johnson argues in the same paper that cloud computing will play an important role in MES implementation, helping to reduce its costs without loss in quality.

Unfortunately, contemporary MES solutions are rather complex software systems, which require significant configuration efforts in order to connect them to existing manufacturing and management systems. This problem is partially caused by strictly hierarchical organization of factory information systems, insufficient description of the system modules and their closed proprietary nature. The concept of Open, Knowledge-Driven Manufacturing Execution System (OKD-MES) is being developed to reduce the complexity and cost introduction and maintenance of MES in companies, while keeping or expanding benefits of MES solutions.

OKD-MES aims to exploit synergy of embedded devices with knowledge-driven service-oriented architecture (SOA) for MES system itself and factory shop floor beyond it. Embedded devices are providing a platform to develop loosely coupled, interoperable and reusable services on factory shop floor level, removing the gap between devices and MES levels of automation hierarchy. Together with knowledge-driven approach applied to OKD-MES services, dynamic reconfiguration of system becomes possible, making system adaptable to changes such as introduction of new tasks, changes in equipment and other. Development of OKD-MES is pursuing a goal to create open, dynamic and smart automated manufacturing environment.

Architecture for OKD-MES is based on five layers: Physical (PHL), Representation (RPL), Orchestration (ORL), Visualization (VIS) and Interface (INT). PHL is closely related to shop floor equipment and corresponding embedded devices. It aims to combine real-time control of devices with capabilities to expose its functionalities as services, and to provide access to description of such services. RPL is being a knowledge repository for the system. This layer contains the model of real world, and provides access to it on demand by other layers in OKD-MES. RPL model of the real world is very dynamic and is loosely coupled to PHL. The layer which may provide actual MES functionality is ORL. It is responsible of the use of services available in the system. Service orchestration and service composition are key functionalities to this layer. Getting information from representation layer ORL is capable to facilitate decisions required for adequate control of the system. As control in such case is based on services and service availability is dictated by devices (PHL) and depicted in RPL, these three layers are capable to provide the core functionality of the system. Two other layers are dedicated to make the system interoperable with humans and other systems, exposing core functionality of OKD-MES to rest of the world.

It is expected that such solution may introduce higher reusability of the shop floor functionality, reduce time required for system configuration, simplify introduction of new equipment or products. OKD-MES even may improve robustness of the system, provided that redundancy in shop floor equipment exists. Integrating the components of contemporary factory in a loose manner and openness to external world should reduce the cost of MES solution, while accent on knowledge may provide capabilities to further automation of manufacturing processes.

Next section will provide overview of background for development of OKD-MES. Concept of OKD-MES will be discussed in details in section 3. In the section 4 the architecture of proposed solution will be described. Separate layers would be provided in subsections of the fourth section. Section 5 will conclude the chapter, highlighting discussion points and defining the future work.

2 Background

This section will provide a concise background for the chapter. The technologies and concepts related to development of OKD-MES, such as Service orientation, web services, their integration and knowledge representation will be overviewed.

2.1 Service orientation and Web Services

Service orientation is a design paradigm based on the concept of services as the modules of functionality. Erl (2007) describes eight principles of service orientation:

- Standardized service contract
- Service loose coupling
- Service abstraction
- Service reusability
- Service autonomy
- Service statelessness
- Service discoverability
- Service composability

These eight principles of service orientation are expected to achieve seven strategic goals of the paradigm. Three out of these goals, such as increased ROI, increased organizational agility and reduced IT burden, are benefits per se. Other four strategic goals are increased federation, increased intrinsic interoperability, increased vendor diversity options and increased business and technology alignment, are enabling the benefits. Principles and strategic goals of Service orientation are presented in Fig. 2.

The most common implementation of SOA currently is according to the Web Services (WS) concept. The concept of SOA interaction which may be implemented as WS is depicted in Fig. 3. As can be interpreted from Fig.3, the Web Service exposes some functionality, once some entity compliant with WS architecture needs to access the functionality provided by WS. The entity such as Service Broker (Fig. 3), is capable to provide information required for connection of service consumer to service provider.

There exist several specifications and architectures to develop interoperable WS in contemporary computational systems. Most prominent among them are RESTful architecture proposed by Fielding, WS-* specification by OASIS. WS-* specification for services provides full palette of standards to implement Web Services. This is advantage and disadvantage same time. Comprehensiveness of WS-* sometimes may lead to higher overhead and complexity of the systems. Especially it is an issue for development of a system as it increases its cost. More simple solution, which is based on HTTP protocol – RESTful Web Services were defined by Fielding (2000). RESTful services are based on HTTP methods and are following the Create, Read, Update and Delete (CRUD) model, applied to computational resources. RESTful services are generally missing a mature standard for their technical and semantic description similar to SA-WSDL for WS-*.

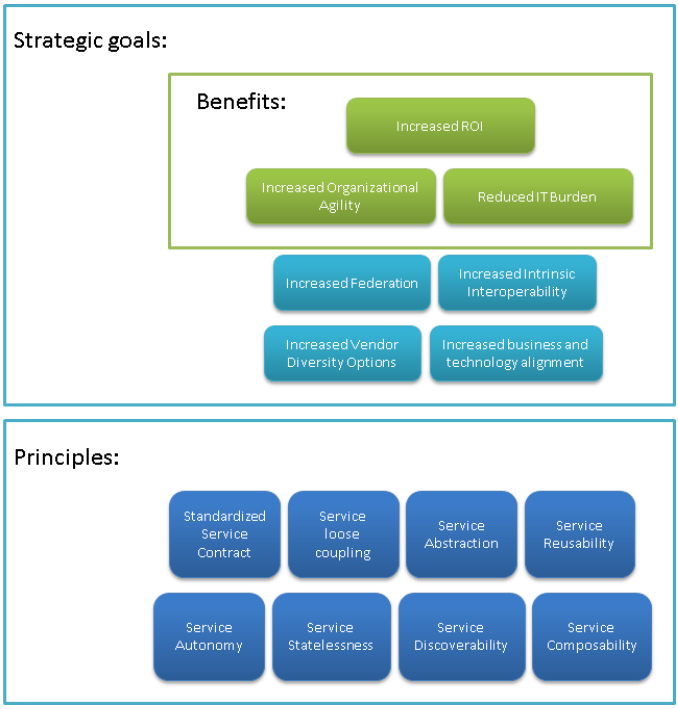


Fig. 2. Principles and goals of service orientation

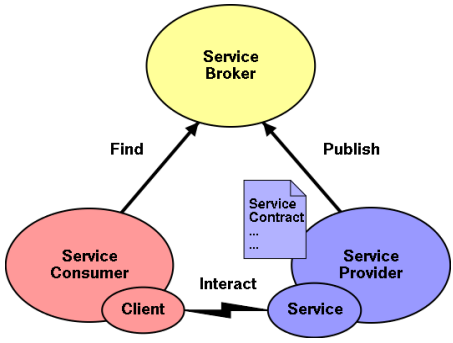


Fig. 3. Interactions in SOA (Haas, 2003)

Web Services are steadily advancing towards factory shop floor devices using expanded capabilities of embedded systems. Web service enabled devices are currently commercially available (S1000). More WS enabled devices are being in development including RTU development which is being conducted within the eScop Project. The eScop RTU is intended to provide capabilities to implement RESTful WS on device, and as well to have some basic commonly required services to support configuration of RTU, data browsing and subscription to it, etc.

2.2 Web Services integration

One of the strategic goals of service orientation, which enables important strategic benefits, is intrinsic interoperability of services. This strategic goal may be achieved if principles of service orientation are being followed. In this subsection principles of service composition, service discoverability and service contract will be discussed.

Possibility of service composition, which enables higher reusability of services and hence efficiency of the system, is very important in scope of OKD-MES. This importance is caused by a need to implement multiple separate and sometimes overlapping functions in order to provide full MES functionality. Unfortunately technical possibility is only necessary, but not sufficient condition for WS integration. Besides technical details the business logic related knowledge is required to define which services should be aggregated in compositions to achieve certain goals.

First of all, to be capable to define which services to integrate, data about service availability needs to be obtainable. Here service discoverability principle is required. Moreover the service should be described from both technical and business logic points of view, for which a service contract has to be applied. Based on the data obtained through service discoverability and service contract and using some models, the service composition may be reasoned, using the Knowledge Representation.

2.3 Knowledge Representation

One of the wide and comprehensive description of Knowledge Representation (KR) was provided by Davis et al. (1993). Generally in application to OKD MES, KR is mainly dedicated to provide a capability for encapsulation of information about the system and its co-dependencies. There exist several formalisms to develop KR, most important of them are semantic nets and ontologies.

Among different implementations formalisms available for knowledge representation the Resource Description Framework (RDF) based family of languages is being most widely exploited (Kalibatiene and Vasilecas, 2011). RDF representation is built of subject-predicate-object triples. More formalization is provided in RDF using RDF Schema (RDFS) and Web Ontology Language (OWL). More complicated formalization leads to wider reasoning capabilities at price of limitations of representation. Each valid OWL representation is also valid RDF, but not vice versa. SPARQL Protocol and RDF Query Language (SPARQL) is a dedicated technology for manipulating the data in RDF based languages.

The possibility of RDF to OWL compatibility is an important benefit of this language family, as level of formalization may be adjusted without a need to change technology stack significantly. For example, less formalized knowledge representation may be used once the structure of knowledge is not clear. Once more information about knowledge domain is accumulated, more complex representation may be introduced to the whole system or some part of it. As well complexity of the data representation may be leveraged by creation of domain ontologies as it is proposed in the chapter *Ontology-based modeling of manufacturing and logistics systems* of this book.

Overall Knowledge Representation together with Web Services concepts are successfully used for manufacturing domain. For example Ramis et al. (2014) offered an approach to use such combination for integration of an industrial automation system. In (Puttonen et al., 2013a) and (Puttonen et al., 2013b) the authors argue about the synergy of KR and WS for representation of a dynamic automation system, and even composition of factory automation processes.

3 OKD-MES Concept

The goal of development of OKD-MES is to provide Open solution, which follows Knowledge Driven approach to implement MES. Considering first term "Open" in this work it has both meaning of being open source solution, open to be studied, modified and distributed. Also it means open, from the perspective of Open Architecture, e.g. the OKD-MES should be open to addition or replacement of modules developed by community. Second part of definition is "Knowledge Driven". Open Knowledge-Driven MES is being developed to be capable to understand the data within the system and hence to be capable to infer additional information based on reasoning mechanisms within OKD-MES.

3.1 Openness

Being developed as an open solution OKD-MES is limited to be based on technologies, standards and components, which may generally be assumed as open ones. Beyond openness from point of view of rights, OKD-MES architecture should be open for modules extending its functionality. For this part the concept of Core MES functionalities and complimentary modules discussed in (Johnson, 2010) may be implemented (Fig. 4). Considering the openness enabled by service orientation it is reasonable to use Web Services in order to encapsulate functionalities of core and complimentary modules of the OKD-MES.

Open Architecture implies modularized and loosely coupled structure. This values maybe provided by SOA. From perspective of service orientation the parts of the system should be developed as interoperable services which can be aggregated in complex service compositions which will provide higher level functionality. Selection of a standard for implementation of WS is rather arguable point. RESTful Web Services described in section 2 are used for development of OKD-MES due to their simplicity, lower communication overhead, development costs, and general acceptance of this architecture in enterprise information systems. The disadvantages of missing standardized formal descriptions of services can be leveraged on the level of knowledge representation.

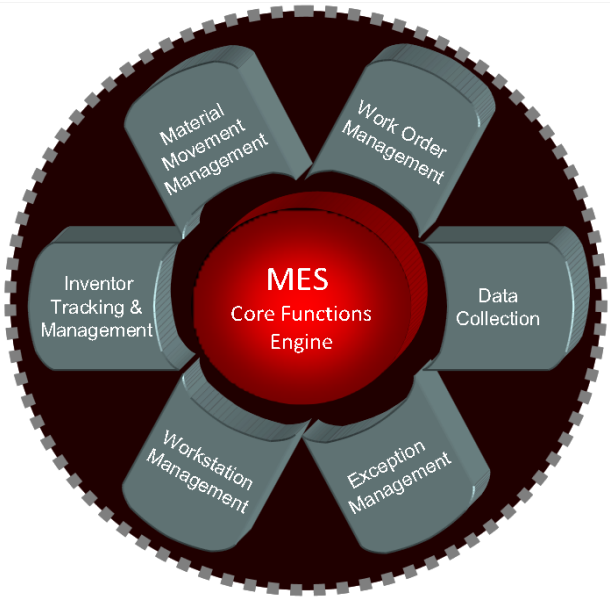


Fig. 4. MES Core and Complimentary functions (Johnson, 2010)

3.2 Knowledge Representation in OKD-MES

Once the requirement of openness is fulfilled in OKD-MES and it is implemented as an expandable set of interoperating modules, to foreground comes the issue of collaboration between corresponding services. Collaboration of the services requires a model in which both technical and business determined parts of contract are represented. Considering extremely broad variety of business functionality available in contemporary manufacturing systems, it becomes obvious that a flexible and expandable solution for such model is required. Moreover this solution has to address large and diverse sets of data, hence has to provide some flexible mechanisms for processing this data before exposing it to user. Knowledge representation discussed in previous sections can provide such properties.

Considering a need of flexibility of the representation, including a possibility to dynamically modify the structure of data and vary the level of details, RDF languages family is a preferred candidate to be used. A Manufacturing System Ontology (MSO) is presumed as a core of knowledge representation. This ontology aims to provide a structure of data required to describe equipment and services available in the system, manufacturing and logistics of processes, visualization screens.

MSO is to be populated with instances of a certain manufacturing system when OKD-MES is configured and partially in run time of the system. Appearance of new services or equipment in the system may trigger a mechanism to update the data in ontology to keep there a relevant representation of a real world. This task can be automated to certain extent, but required semantics still has to be provided to the system at some point.

Semantic enrichment of equipment in manufacturing line may be provided by the manufacturer. Other information about shop floor layout, e.g. connections between equipment and possibly some customization for it has to be introduced already on system deployment. Description of visualization screens and processes have to be introduced when the OKD-MES is being connected to a manufacturing system, using already existing knowledge about equipment and connections between them. It is beneficial to keep the named domains loosely coupled between each other, as it reduces the complexity of knowledge each OKD-MES user must have in order to configure a system.

3.3 Core and complimentary components

Manufacturing Execution Systems are built on the top of factory shop floor level, hence the connectivity of shop floor to MES is one of the core functions that OKD-MES should provide. Considering availability of embedded systems for the shop floor devices, it is possible to use WS to enable such connections. Hence the functionalities of connectivity with the shop floor equipment may be encapsulated into a Physical Layer.

As was discussed before, the information about representation of the system in general, and service in particular, should be persisted to enable functionality of Knowledge Driven solution. This core functionality may be encapsulated in the form of a Knowledge Base and be a part of Representation Layer. Also this layer should include functionalities related to manipulation of the KB, from introduction of new instances, to modification of the models, and reasoning to maintain coherent real world representation.

Considering service oriented nature of OKD-MES, another ubiquitous functionality based on the principles of service orientation is the capability of system to combine services in service composition and execute such compositions. Generally there exist two main approaches to execute service compositions, namely: orchestration and choreography.

Orchestration implies a central entity, responsible for coordination of the services, while choreography distributes this functionality between all services in the system. Taking into account a need of a system to absorb new modules, it is reasonable to reduce complexity of the services itself, especially in case of application of resource constrained embedded devices from the factory shop floor. Hence the execution of composed services is delegated in the system to Orchestration Layer. This layer may also include some form of composition tool, which based on the knowledge from Representation Layer and user input should generate the executable descriptions of service compositions.

Interactions with external systems may introduce a need to create adapters and restrict access to some parts of the system for security reasons. This functionality should be addressed by the Interface Layer.

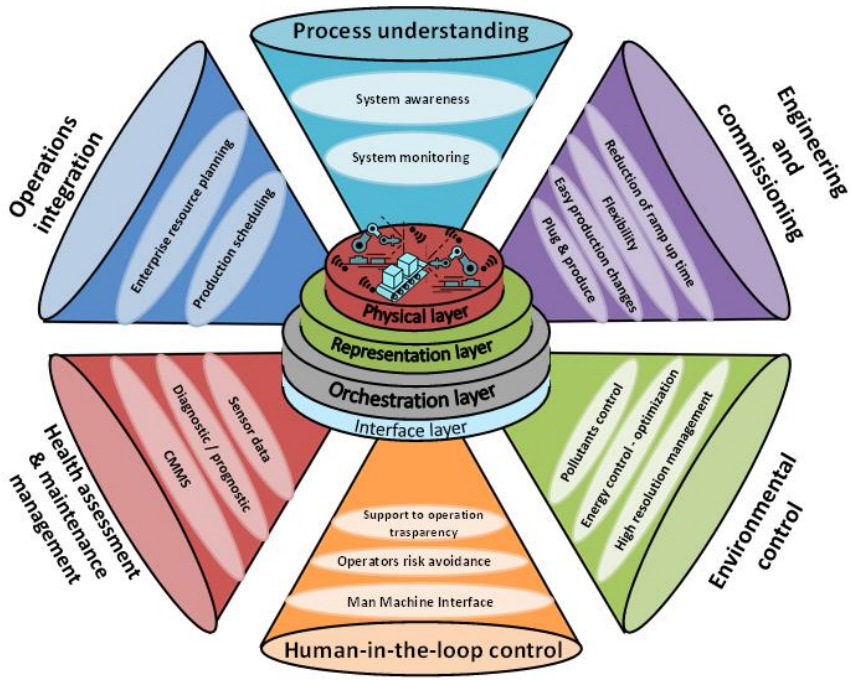


Fig. 5. General eScop Architecture

Other functionalities which the information system should provide are to be developed as complimentary modules. The concept of layered, modular, knowledge driven system is depicted in Fig. 5.

Another ubiquitous functionality which may be assumed to be a part of OKD-MES core is visualization. Being the only way of human interaction in the system most of other parts of the OKD-MES (both core and complimentary) require functionality provided by visualization in order to be used. Possibility of development of flexible, dynamic, knowledge based visualization system is also assessed in this chapter.

4 Architecture

In current section the concept of OKD-MES is discussed in more details, the architecture for such system will be presented, functionality and possible realization of core components will be demonstrated.

Architecture of OKD-MES core includes five main components (Fig. 6): Physical Layer (PHL), Representation Layer (RPL), Orchestration Layer (ORL), Visualization Agent (VIS), and Interface Layer (INT). Integration of OKD-MES core with shop floor equipment is provided using functionality of PHL. Complimentary functions of OKD-MES are expected to be implemented as services and/or clients of OKD-MES core functionalities. Access to such core functionalities is provided via Interface Layer in order to provide required level of access to core.

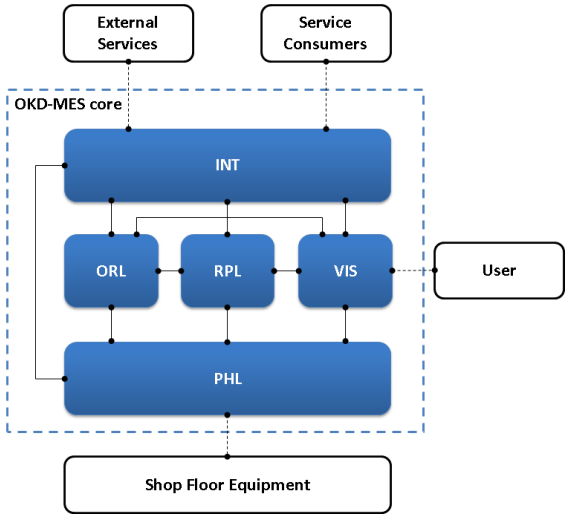


Fig. 6. OKD-MES core architecture

Finally services implementing the core or complimentary functions of MES are able to use a Visualization agent to provide user interface for the system.

4.1 Interactions between layers

As it was already mentioned, interface layer should be capable to define level of access of external service or user and enable this access to underlying services of OKD-MES core. It is also possible to use adapters in interface layer to connect the MES to providers and consumers which are not implementing RESTful WS or even are not services at all.

Physical layer should be capable to provide the information required to create representation of shop floor devices in the Knowledge Base of RPL. This information must include shop floor device description from manufacturing point of view as well as from service perspective. Directly or indirectly the information about equipment status is to be exposed by PHL to RPL.

Orchestration Layer should be capable to execute service on PHL in order to orchestrate service compositions. On other hand it should be capable to read some data from shop floor equipment if it is required for process definition. In some cases it may be required to subscribe for notifications from PHL in order to execute the service composition efficiently.

The definition of the executable processes in ORL may and should be abstracted from real service implementations and be concentrated on functionalities for the reasons of robustness and flexibility of the approach. Hence the mapping of these abstract services to their real implementations should take place. This on the mapping information related to service invocation should be provided. All required information connecting the syntactic and semantic description of the services is persisted in RPL. This mapping functionality is one of the main interactions between ORL and RPL. As

well some information about process execution and status of the system is an important part of interaction of these two layers.

Interactions between Visualization Agent and RPL are required to generate the visualization screens on demand. Information about configuration of the screen including data points for subscription in other layers should be provided by RPL.

Interactions between Visualization Agent and PHL are to be mainly of subscription format. E.g. some information explicitly available in the device may be required. All interactions directed from visualization as input should be developed in a form of executions of certain services in the system.

4.2 Physical Layer

Architecture of physical layer is based on the concept that device functionality should be encapsulated in a service. The data from device should be accessible through the WS interface, the operations on devices may be invoked as services. For obvious reasons PHL should provide capabilities to discover and describe the service in order to enable loose coupled integration in overall system. Moreover some basic services as subscription should be implemented on Physical layer to enable event oriented behavior of the system.

In simple most case interaction Physical layer may provide simple read/write capabilities for a simple I/O module, including mechanisms for discovery, description and subscription. Considering that control is based on inputs and outputs of the system writing data in I/O module may indirectly trigger an operation.

A more efficient and flexible solution is to provide an access to a runtime core. In the control programs running in PHL device runtime core events may be defined. The data consumers may subscribe to such events and be connected to more abstract data than one available from a simple I/O module. Also possibility to invoke the programs in PHL device directly using some data from service call is a more flexible approach.

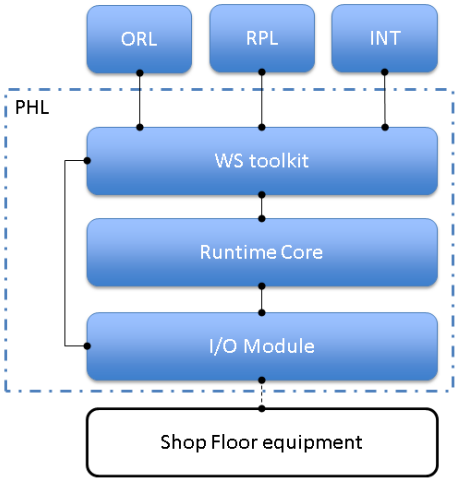


Fig. 7. Architecture of the Physical Layer

The architecture of PHL developed for OKD-MES includes three main modules (Fig. 7.): Simple I/O module, Runtime Core and WS Toolkit. WS Toolkit enables web services functionality and may be connected both to Runtime Core and I/O Module. Besides, already named service oriented functionality WS Toolkit may provide an interface for configuration of PHL. I/O Module is implemented as a bridge to real inputs and outputs of a device. Runtime Core is an architectural block on which the control logic is being executed. In case if PHL device is being implemented as a bridge to some legacy device, the Runtime Core may be virtually omitted.

4.3 Representation Layer Architecture

Architecture for Representation Layer is developed based on the services it will provide for other layers. Most of the services of RPL will have interaction with MSO such as CRUD (create, read, update and delete) operations on the Ontology model. Depending on the service consumer, the services can be separated into four modules in order to facilitate the development and maintenance, namely, Device Registration Module, Visualization Provider Module, Service Handler Module and Manager Module. These modules all need some internal functions in order to interact with MSO which is stored in an Ontology database. The architecture of RPL is depicted in Fig. 8. The components of the architecture can be explained as follows:

Manager Module: It provides configuration interface, model browser/editor and CRUD operation on MSO for IT staff or authorized users.

Device Registration Module: A service module to register or unregister eScop devices in the MSO through device catalogue. It enables access to device existence and their capabilities in the network. Mainly works with Physical domain in MSO.

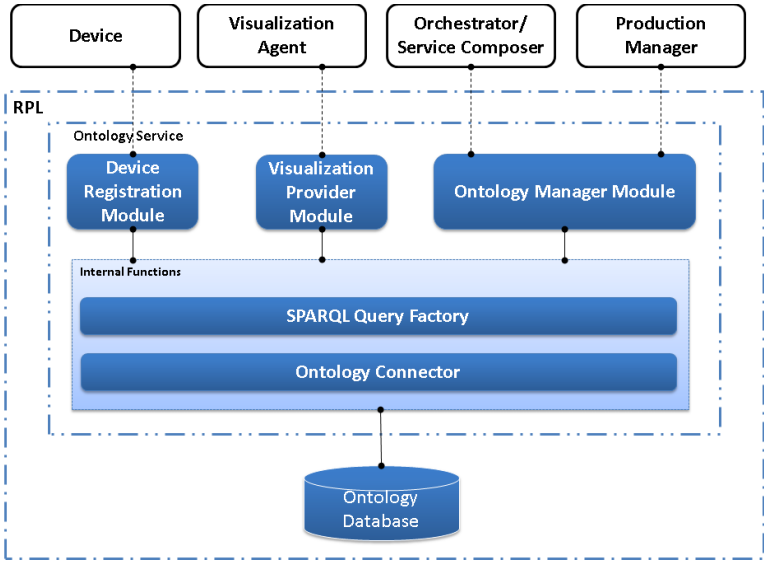


Fig. 8. Architecture of Representation Layer

Service Handler Module: A service module to communicate with Service Composer, Orchestrator Services and Production Manager. The module should expose services to provide information requested from the Service composer, receive update requests from Orchestration service and receive update requests from Production Manager. The module must handle description of the services, configuration and status of the system, SPARQL over HTTP could be used for flexibility of the approach, but it is worth to mention it is necessary to introduce authorization and verification mechanism for SPARQL over HTTP in order to keep data integrity of the MSO.

SPARQL Query Factory: It is basic common function supporting generation of SPARQL queries.

Ontology Connector: Basic Common function to execute SPARQL queries and return the query result.

4.4 Orchestration Layer Architecture

Architecture for Orchestration Layer is developed considering its two main functionalities: orchestration and composition. A Service Composition module is enabling the process of creation of complex service compositions. Creation of service composition requires knowledge about existing services which may be provided by RPL. The composed services are to be returned back to RPL to be persisted in the system. In most cases on service composition interactions with a user would be required to make ultimate decisions which services to use and in which patterns.

Second module of ORL architecture is Service Orchestrator. This has to enable execution of service compositions and providing information about the execution for other parts of the system. Orchestration of service compositions means that Service Orchestrator should include orchestration engine as well as other components required for interactions with other systems. Generic view on ORL architecture is presented in Fig.9.

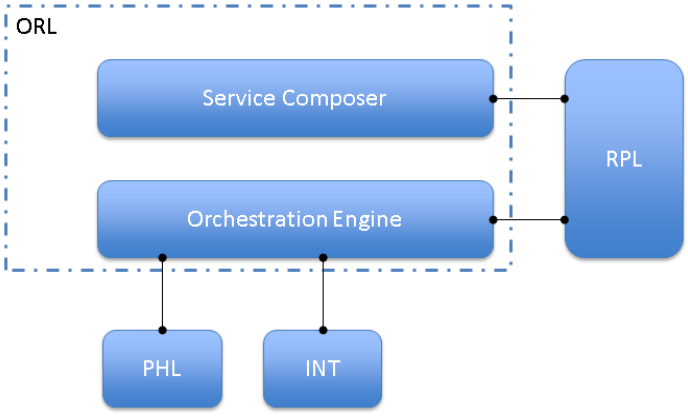


Fig. 9. Architecture of Orchestration Layer

4.5 Visualization Layer Architecture

Visualization Layer has to provide a visualization agent capable to generate visualization screens based on a screen descriptions persisted in RPL. Moreover, considering the factor of ubiquity, implementation of a browser as a visualization client is presumed. The general view of the architecture of Visualization Layer is presented in Fig. 10.

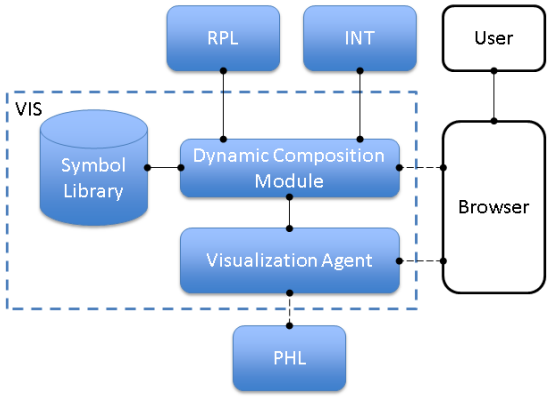


Fig. 10. Architecture of Visualization Layer

Considering a need to provide visualization for dynamically composed screens, it is required to have a repository of symbols to which visualization descriptions may be linked. This symbol repository is presented as Symbol Library (Fig. 10). Dynamic Composition Module is to be using this symbol repository in order to generate visualization descriptions which can be transformed to browser readable format by visualization agent. Additionally Dynamic Composition Module of VIS has to create required subscriptions to data in the other parts of OKD-MES in order to represent in on visualization screen. Finally Visualization Agent is a part of architecture which has to transform the visualization screen description to a web page of some form, which can be displayed in browser being connected to a real data in the system.

5 Summary

This chapter provides a concise review of decisions made in development of Open Knowledge Driven Manufacturing Execution System. It includes a description of expected benefits of the presented approach. The principles and concept which are suggested to use in order to achieve the benefits are described. Finally architecture of OKD-MES is described.

More details of the architecture of the modules will be revealed along implementation of OKD-MES core functions as well as during expanding the system with modules responsible for certain MES functions. Also application of the developed OKD-MES may provide more feedback about design decisions presented in this chapter.

References

Accenture (2010). Manufacturing Execution Systems. http://www.accenture.com/us-en/Pages/insight-achieving-high-performance-manufacturing-execution-systems-full.aspx. Accessed 2015.II.26.

Davis, R., Shrobe, H., & Szolovits, P. (1993). What Is a Knowledge Representation? *AI Magazine*, 14/1: 17.

Erl, T. (2007). SOA: Principles of Service Design. Prentice Hall.

Fielding, R.T. (2000). Architectural styles and the design of network-based software architectures. University of California, Irvine.

Haas, H. (2003). Designing the architecture for Web services. Presented at the WWW2003, Budapest, Hungary. Retrieved from http://www.w3.org/2003/Talks/0521-hh-wsa/slide5-0.html. Accessed 2015.II.26.

Johnson, F.K. (2010). Future of Manufacturing Execution Systems: The Brave New Modular World of Manufacturing Intelligence. *Review of Management*, Vol. 4.

Kalibatiene, D., & Vasilecas, O. (2011). Survey on Ontology Languages. In: Grabis J., Kirikova M. (eds), *Perspectives in Business Informatics Research*: 124–141. http://link.springer.com/chapter/10.1007/978-3-642-24511-4_10. Accessed 2015.II.26.

Kletti, J. (2007). Manufacturing Execution Systems (MES). Berlin; London: Springer. Retrieved from http://www.books24x7com/marc.asp?bookid=30973

McClellan, M. (1997). Applying Manufacturing Execution Systems. CRC Press.

MESA (2014). International - Home (n.d.) http://www.mesa.org/en/index.asp. Accessed 2014.X.3.

Puttonen, J., Lobov, A., & Lastra, J.L.M. (2013a). Maintaining a Dynamic View of Semantic Web Services Representing Factory Automation Systems. *20th International IEEE Conference on Web Services (ICWS)*: 419–426. http://doi.org/10.1109/ICWS.2013.63. Accessed 2015.II.26.

Puttonen, J., Lobov, A., & Lastra, J.L.M. (2013b). Semantics-Based Composition of Factory Automation Processes Encapsulated by Web Services. *IEEE Transactions on Industrial Informatics*, 9(4): 2349–2359. http://doi.org/10.1109/TII.2012.2220554. Accessed 2015.II.26.

Ramis, B., Gonzalez, L., Iarovyi, S., Lobov, A., Lastra, J.L.M., Vyatkin, V., & Dai, W. (2014). Knowledge-based web service integration for industrial automation. *12th IEEE International Conference on Industrial Informatics (INDIN'2014)*: 733–739. http://doi.org/10.1109/INDIN.2014.6945604. Accessed 2015.II.26.

VDI (2014). Verein Deutscher Ingenieure: Sprecher, Gestalter, Netzwerker (n.d.) https://www.vdi.de/. Accessed 2014.XI.24.

On Service Composition

Dynamic Formation and Orchestration of Service Workflows

Andrei Lobov

Tampere University of Technology, Tampere, Finland

andrei.lobov@tut.fi

Abstract. Service-oriented approach to system engineering pushes to rethink the way, engineers build and maintain systems. Use of Web Service technologies enables to remove rigid connections between software components, to reassemble them dynamically at a run-time, according to the actual needs of applications. Such an approach can in some cases provide additional necessary functionality, which may not have been envisioned at the design time. This chapter presents an approach to orchestration allowing dynamic formation of service hierarchies, in in line with production needs, which allows tracking on all existing service workflows. Hence the locus of control for the overall system is kept. The use of eScop Manufacturing System Ontology (MSO) for orchestration purposes is also proposed, to keep the track on orchestration workflows.

1 Introduction

As the computational and communication capacities of embedded devices have been constantly growing, it becomes possible to increase performance of computations and communications at their level. In principle it makes possible to run service composition and orchestration tasks at this level.

At the same time, the field of industrial automation is conservative. Due to manufacturing systems are often build for the manufacturing of concrete products without or with controlled customization, the benefits of dynamic system reconfiguration for manufacturing systems may not be apparent for the factory owners. However, even in the cases of mass production, the potentials of service-oriented architecture (SOA) can be seen within monitoring and optimization of already running systems.

In order to support the acceptance of SOA in manufacturing systems domain, the service-enabled devices and support services have to be easily discoverable and composable at all the levels of so called the automation pyramid. Also, the service composition methods, languages and techniques should be understandable and runnable at all the levels ranging from factory floor up to ERP systems.

This chapter introduces a background on service composition. It outlines five most common problems that researchers face in order to improve dynamic and runtime service composition. It also describes the suggested approach to orchestration, followed by discussion on the use of the eScop MSO, to support service orchestration.

2 Background

There are two main approaches to service composition. One is the service orchestration and another one is service choreography. The orchestration supposes a centralized approach to service composition where the central entity (orchestrator) is responsible for controlling the whole process of service execution. In the case of choreography, each application component (a service) provides service interfaces and has the information about its peers and their services, which it may invoke, as a result of its service call. Thus, no central entity is required for making service choreography.

There are dedicated languages in order to describe each of service composition approaches. For orchestration purposes Business Process Execution Language (BPEL) (OASIS, 2007) can be used. The choreography of Web Services can be described using Web Services Choreography Description Language (WS-CDL) (W3C, 2005).

In its extreme, the orchestration has a weakness of having a single point of failure - process or workflow orchestrator. If it stops, the whole process is halted. It also requires bigger amount of message interexchange in comparison to choreography. On the other hand, in choreography it becomes easier to lose a locus of control, as devices or computers holding the services do not necessary report to any common observer so that the (un)successful execution of a composite service is known often only when the final result is received (or not received). Also choreography makes system validation more difficult, as it is stated in (Mostarda et al., 2010).

An ability to validate is connected to the observability of system state. Since in service-oriented systems several service workflows can be executed simultaneously, the state of each workflow and overall system state as a superposition of workflow states should be measurable. The term *decentralized orchestration* is used to describe possibly simultaneous execution of workflows at separate locations. The workflows can be also divided into partitions that can be orchestrated independently.

First attempts for describing decentralized orchestration began at the same time as abovementioned standards emerged. Binder et al. (2006) introduced special service invocation triggers to formalize interconnections between the services, which help to create so called decentralized workflows. The triggers may store the information where to send the data of service execution thus avoiding communication through the central orchestrator. Mostarda et al. (2010) have proposed dedicated units, called manager and leader, supported by a set of backups. A leader has a workflow skeleton and knows the state of each workflow instance. The manager holds finite state machine built from information in local state machines. If manager information gets outdated, the leader updates it. The backups meant for recovery processes in the cases when leader fails.

The entire frameworks are developed for service orchestration. As some global task can require a set of services to be invoked, finding the right services can be seen as building a coalition. Rubio-Loyola et al. (2011) presented service-centric orchestration protocol, supported by the framework containing dynamic planner, which is serving distributed orchestration components.

Fdhila et al. (2012) proposed a framework for partitioned orchestration of Web Services. It defines process partitioning function, where resulting partitions can com-

municate with each other through dedicated communication patterns. That work also specifies how to handle the propagation of a change on centralized orchestration model to dedicated partitions.

Jaradat et al. (2013) described architecture, which claims to move computations closer to services in a workflow. The workflow is partitioned and special proxy services are introduced in architecture, to maintain state and data related information for the orchestrated services. The orchestration service does not access (all) services directly, but rather invokes them through the proxy services. The proxy service can be seen as some sort of a wrapper for a set of more basic services.

From those few papers referred above, it is already possible to highlight main approaches that enable handling the service composition process in more efficient, observable and controllable way. The main questions based on referred works can be formulated as follows:

1. What should be information structure to store data on services (location, parameters, users of results…)? (e.g. "invocation triggers");
2. What should be main components and interactions between those in order to allow distributed orchestration? (e.g. leader, manager, backups);,
3. What should be orchestration framework to allow service orchestration? (e.g. introducing service-centric approach, with dynamically forming "coalitions");
4. How formally partition the workflows?
5. How services can be wrapped? (e.g. introduction of proxies).

While these questions are answered to some extend in corresponding publications, they still lack a common extendable framework having all of those major properties implemented for successful dynamic service composition.

The eScop project develops an architecture allowing service orchestration. Next section shows how the listed questions can be addressed.

3 Approach

The eScop architecture is described in other chapters of this book. In brief, it contains three core layers: orchestration, representation and physical layer. In addition there is visualization layer, which forms operator interfaces, or interfaces for other users of monitoring applications. In addition, services for MES functions can be seen to form actual application layer. The layers and their components set up the answer for the second question listed in the previous section. One of the important layers is the representation layer, which stores information on available services and workflows. Thus, it should contain required structures supporting service composition.

One of the key services located at the orchestration layer is *ServiceComposer*. The workflow creation starts from it. The *ServiceComposer* receives as an input product or production needs and translates those into a workflow. This service interacts with representation layer service *RPLService* to get a list of services required to fulfil the product needs, then it composes the workflow and distributes it among the *Orchestrators*.

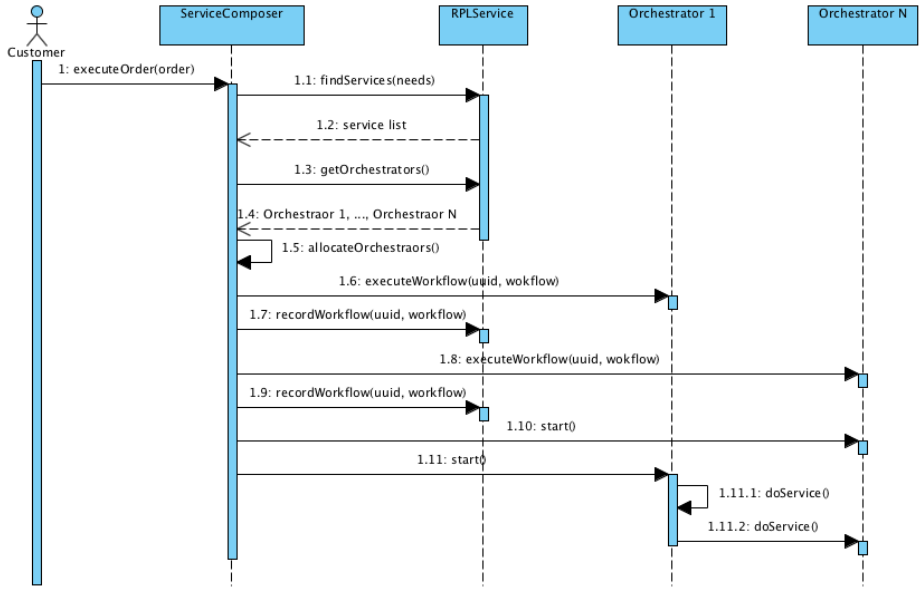


Fig. 1. ServiceComposer assigning dependent workflows to a number of orchestrators

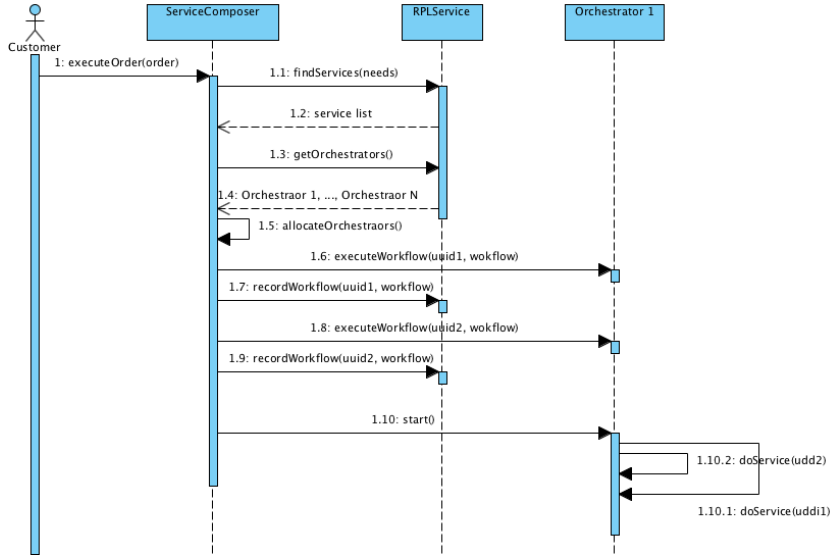


Fig. 2. ServiceComposer assigning dependent workflows to the same orchestrator

Fig. 1 and Fig. 2 depict the UML sequence diagrams, which highlight interactions between the services. There are two workflows assigned to the orchestrator(s). The workflows are dependent, which means that the second workflow is embedded into the first one.

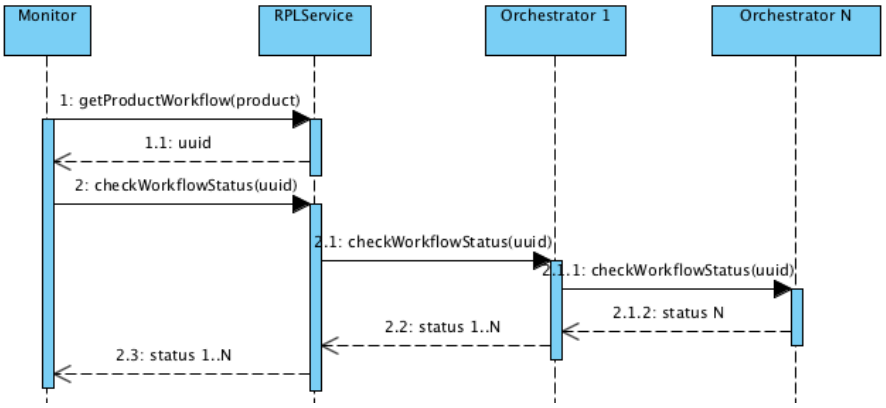


Fig. 3. Monitoring the workflows (pull model)

ServiceComposer based on available *Orchestrators* makes the partitioning of workflows. The metrics to decide which *Orchestrator* to select is based on its CPU power, memory, current load with other workflows execution and distance to the services that should be orchestrated. The distance can be measured as the number of different networks (routers) that have to be passed for reaching a service to orchestrate. The distance is 0, if the service and its orchestrator are located in the same network. The calculation of the distance can be also adjusted depending on the intermediate network properties (e.g. data rate).

In both cases (Fig. 1 and Fig. 2) *ServiceComposer* is the one, which must trigger execution of the workflow. Once triggering, the *ServiceComposer* assigns Universally Unique Identifier (UUID), which is used to unambiguously refer to a workflow. UUID is defined in RFC4122 standard developed by Internet Engineering Taskforce (IETF). It is a 128 bits long string to "guarantee uniqueness across space and time" (IETF, 2005). The UUIDs are stored in eScop Manufacturing System Ontology (MSO) to record the workflows. The recording communicates the status change of workflows – where is a workflow at the moment.

The monitoring of workflow status can follow two basic approaches pull and push. With the pull approach illustrated in Fig. 3, the dedicated *Monitor* application required to know the status of a workflow makes first a request to *RPLService* to check what is an UUID for the product. Then, the *Monitor* requests through the same *RPLService* the status of the workflow with the UUID.

In the push approach, the *Orchestrator* updates on each state change the *RPLService.* This however would increase amount of messages exchanged on each service invocation in each of the workflows. The workflow monitoring approach can be included as a parameter to allow selection of either pull or push models for different workflows in the same system.

In the case of push approach, a listener for model updates should be implemented to specify the parts of the models, which change should result in a notification for subscribed monitors. The monitors while subscribing should specify which parts of the model they would like to listen to.

The monitoring of the workflow status gives important information. It allows checking the use of resources and its timing that can be translated into system performance with respect to a product it manufactures. The monitors can also serve as exception handlers checking for errors and failures to initiate recovery procedure that can be started through *ServiceComposer*.

In order to support described scenarios, the *RPLService* should contain valid models for workflow representation. Next section provides some more details on workflow modelling.

4 Modelling workflows

The eScop MSO already contains a concept of *DiscreteProcess*. This concept has a list of properties including ID, routing, product to be produced and operation precedence list (Fig. 4).

Actually *ServiceComposer* should initialize the ID with UUID. The concept stores information on the product, which will be produced by the process and defines what are the product needs, which can be seen as the routing of the product. The exact orchestration order is defined in operation precedence list.

During runtime, the operation, which has been already performed, can be deleted from product routing, thus representing the status through remaining product needs. This information together with precedence list can unambiguously tell what is the current status of the process. An extra property can be added to the *DiscreteProcess* to tell if it uses pull or push type for monitoring of the ontology model. The push model should contain the list of subscribers, which have to be updated on the status change in process execution.

The requests and updates to *RPLService* are made as SPARQL and SPARQL Update (W3C, 2013) queries. The latter can modify the ontology model. For example, the *Orchestrator*, when asked to report its status, can just execute an UPDATE query on the model and remove from routing already fulfilled needs of the product.

In order to check the operations in the routing, the following query depicted in Fig. 5 should be executed:

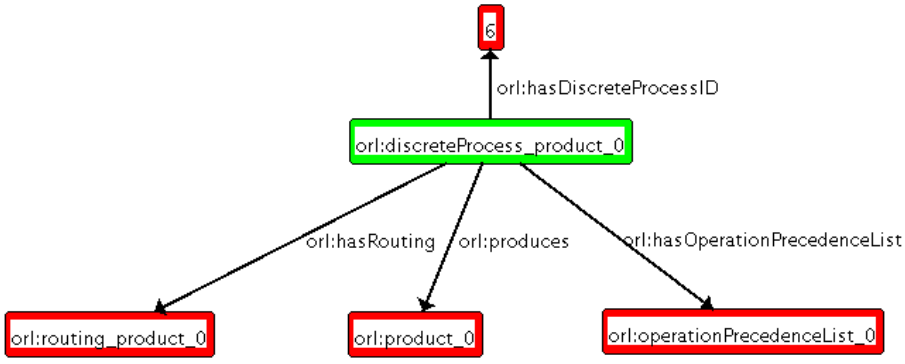


Fig. 4. Outgoing properties for *DiscreteProcess* concept

```
PREFIX mso:<http://www.escop-project.eu/MSO.owl#>
SELECT ?process ?routing ?operation
WHERE {
?process mso:hasRouting ?routing .
?routing mso:hasOperation ?operation .
}
```

Fig. 5. SPARQL query to list processes with their routing and corresponding operations

```
PREFIX mso:<http://www.escop-project.eu/MSO.owl#>
SELECT ?routing ?operation ?precedence ?precedenceList
WHERE {
?precedenceList mso:hasOperationPrecedence ?precedence .
?precedence mso:goesFrom ?operation .
?routing mso:hasOperation ?operation .
}
```

Fig. 6. SPARQL query to list operations for which the precedence list and routing are defined

Another query can be executed to check which operations are left in the precedence list and which are also remaining in the routing (needed for the product). This query is presented in Fig. 6.

The result of the query can be used to report the status of a workflow showing what operations still have to be executed. Also the results of query shown in Fig. 6 can be intersected with the operations in precedence list – the complement of this intersection will give operation list that were already made on a product.

5 Conclusions

To resume, it is worth to note some novelty of the research and obtained results. In the presented approach the orchestrated services are wrapped using *Orchestrator* services. The workflows are defined by *ServiceComposer* service. The partitioning of services between *Orchestrators* can be decided based on CPU, memory, current load and distance of an *Orchestrator* to orchestrated services. The *Monitors* can subscribe and follow service orchestration processes. These can be also used to implement exception handling. The interaction patterns between the main components were described. Most important element in knowledge-driven system is a knowledge base, which, in this case, is built using web ontology language. The knowledge base can be queried and updated. The main components (services) while requesting or reporting from/to the knowledge base can directly use SPARQL (Update) queries.

The presented approach allows always keeping the locus of control on current state of the system. The overall state of the system is a superposition of the states of its workflows. This information is possible to retrieve with a dedicated SPARQL query.

References

Binder, W., Constantinescu, I., & Faltings, B. (2006). Decentralized Orchestration of Composite Web Services. *Proceedings of the International Conference on Web Services (ICWS'2006)*: 869-876.

Fdhila, W., Rinderle-Ma, S., Baouab, A., Perrin, O., & Godart, C. (2012). On evolving partitioned Web Service orchestrations. *Proceedings of the 5th IEEE International Conference on Service-Oriented Computing and Applications (SOCA'2012)*: 1-6.

IETF (2005). A Universally Unique IDentifier (UUID) URN Namespace, Network Working Group, Internet Engineering Taskforce. http://tools.ietf.org/html/rfc4122. Accessed 2015.IV.30.

Jaradat, W., Dearle, A., & Barker, A. (2013). An Architecture for Decentralised Orchestration of Web Service Workflows. *Proceedings of the 20th IEEE International Conference on Web Services (ICWS'2013)*: 603-604.

Mostarda, L., Marinovic, S., & Dulay, N. (2010). Distributed Orchestration of Pervasive Services. *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA'2010)*: 166-173.

OASIS (2007). Web Services Business Process Execution Language Version 2.0. Organization for the Advancement of Structured Information Standards., https://www.oasis-open.org/committees/download.php/23964/wsbpel-v2.0-primer.htm. Accessed 2015.IV.30.

Rubio-Loyola, J., Merida-Campos, C., Serrat, J., Macedo, D. F., Davy, S., Movahedi, Z., & Pujolle, G. (2011). A service-centric orchestration protocol for self-organizing autonomic management systems. *IEEE Network*, Vol.25/6: 16-23.

W3C (2005). Web Services Choreography Description Language Version 1.0. World Wide Web Consortium. http://www.w3.org/TR/ws-cdl-10/. Accessed 2015.IV.30.

W3C (2013). SPARQL 1.1 Update. World Wide Web Consortium, http://www.w3.org/TR/sparql11-update/. Accessed 2015.IV.30.

Knowledge-Based Production Planning

Identification of Model Structures for Production Planning Procedures

Tomasz Borowiecki[1] and Roman Stryjski[2]

University of Zielona Góra, Zielona Góra, Poland

[1]t.borowiecki@iizp.uz.zgora.pl, [2]stryjski@post.pl

Abstract. This chapter presents specification of computational architecture aimed at representation of knowledge in the area of production planning and control. The proposed solution for the rapid prototyping of models of production planning procedures (in the form of executable CSP/COP scripts) makes the task a less time consuming. The need for expert knowledge (in the models of design process of production planning procedures) is likewise reduced. This approach is addressed especially to software developers interested in the realization of projects in the area of production planning and scheduling functionalities of ERP and MES systems. The focus of discussion herein is on theoretical fundamentals and the most important implementation aspects with regard to the form of models for designing system.

1 Introduction

Production planning and scheduling functionalities are integral part of nowadays ERP and MES systems. From systems architecture point of view these functions in most cases take a closed form, being developed as suitable for solving specific class of production planning and scheduling problems. From software customer point of view, functionalities of production planning and scheduling are a crucial criterion while choosing Production Planning and Scheduling (PPS) system. One of the main requirements is a strict suitability for particular production situation. It has to be emphasized that PPS problems formulation strongly depends on production system and planning process organization, i.e. dependencies involved by subsystems and complex system elements, production flow organization and resources constraints have to be considered. Hence the traditional approach based on software requirements definition, solving algorithms development, implementation, and practical application of developed modules is very cost consuming process (in sense of time of development and in reference to the involved knowledge requirements). From the other hand, production planning and scheduling problems formulations (even in very different types of enterprises) has many common points. From a software developer point of view there is a need for efficient methods enabling construction of freely customizable optimization models for production planning and scheduling procedures.

In our opinion the methods addressed herein should support development process (of optimization models for PPS procedures) in the following areas:

- Formal description of production situation
- Representation and efficient reuse of knowledge in PPS
- Methods for identification/code generation of models of PPS problems (derived from combination of values of parameters in production situations description and knowledge base)
- Computational architecture for effective development of models structure by identification/prototyping/verification and executable code generation

2 Model of Production Planning and Scheduling procedure

Development process of required model and solving algorithms for particular production situations can be treated analogously to problems of development of methods for engineering computations (Agrawal et al., 1993; Papalambros and Wilde, 2000). Many dependencies between interacting elements of production system and complex technological conditions of production flow organization, have to be mapped into feasible model (in sense of obtaining solutions and efficient due to computation time). Possibility of different results of model structure due to using different decomposition strategies, effects the process of constructing feasible models of production planning and scheduling procedure. It courses that development and implementation process seems to be ad-hoc and consuming much of expert knowledge. From the other point of view many planning and scheduling problems, especially considering planning procedures sub problem formulations (as the result of decomposition strategy) seems to be very common for different practical cases. Software development environment based on architecture allowing: represent partial models, validate dependencies between them and forming knowledge representation about feasible structures of prototyped models (Borowiecki, 2007; Borowiecki et al., 2004). Additionally supporting mechanisms for model structure description for transformation to executable code, gives the chance to fulfil required flexibility of development of customized software for production planning and scheduling efficiently.

Due to practical condition of implementation of proposed mechanisms let us assume in the presented approach, that a partial model production planning and scheduling procedure has the form of CSP/COP (Constraints Satisfaction Problem / Constraints Optimization Problem) model. There are many publication confirming efficiency of solving PPS problems (Baptiste et al. 2001; Fromherz, 2001; Bradwell et al., 2000; Borowiecki et al., 2001; Borowiecki and Banaszak, 2000). Solution of the problem formulated as CSP/COP is derived basing on the controlled search process. Decision variables values evaluation in model is obtained due to search strategy and considers constraints propagation (narrowing variables' domains). In many CSP/COP systems (e.g. Mozart) model of the problem has the form of: variables declaration, constraints definition, search strategy declaration (and/or implementation). More information about CSP/CSP solver can be found in (Schulte et. al., 1998; Gecode, 2009; Hooker, 2004).

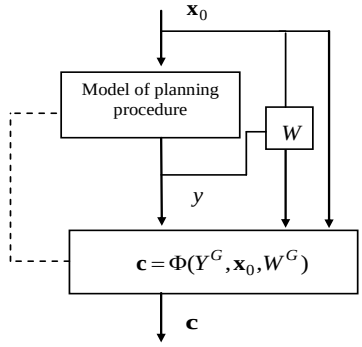


Fig. 1. Role of planning procedure model

From the point of view of architecture of ERP and MES systems, functionalities of production planning and scheduling take form of application integrated with main system as a module or a service. Form technical point of view planning module implementation can be treated as standalone architecture controlled by the ERP/MES system. In Fig. 1. $\mathbf{x}_0$ stands for data acquired form ERP/MES system. Considering the basing on the feasible CSP/COP model of planning and scheduling problem and problem solver engine, it is possible to find a feasible solution y or a set Y of feasible solutions of the modelled problem. We consider definition of performance index function $W = \eta(\mathbf{x}, Y)$ also. In practice definition of performance index function determines iterative execution of search process (dashed line). Let us assume existence of global G constraints (involving the elements of vector $\mathbf{c} = \Phi(Y^G, \mathbf{x}, W^G)$ as a system control).

3 Partial models (models of sub-problems)

Let us assume the partial (sub-problem) model takes the form similar to the model of planning procedure introduced earlier (Fig. 1). Let $\mathbf{x}_p$ be a vector of input attributes (parameters) of sub-problem's model p. Consequentially $w_p = \eta(x_p, \mathbf{y}_p)$ is locally implemented definition of performance index function, $\mathbf{y}_p \in Y_p$ is the one of the feasible solutions Y_p.

Model of the sub-problem represents a stage in decision making process and generally corresponds to some algebraic expression, an algorithm of values declaration/transformation, numeric procedure or as in this case CSP/COP model of locally defined sub problem (batching, load balancing, resources allocation, activities scheduling, etc.). Stages of decision process execution can be represented by the elements and structure of the model of production planning procedure (Fig. 2).

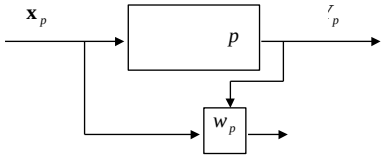


Fig. 2. Model of sub problem

Due to complexity of the practical production planning and scheduling problems most of practically implemented engines corresponds with models derived as composition of interconnected sub-problem models (as the result of implementation of assumed decomposition strategy derived for some class of planning and scheduling problems). The problem lies in the traditional development path: problem formulation, problem decomposition, model construction and solving algorithms development, testing and assumptions correction is generally not effective in this specific area. Hence it takes place when many preliminary activities (planning problem class and technological conditions of production flow identification, considering constraints and dependencies generated from system and production flow organization) are required. So traditional approach seems to be inevitable, as it is used by many researchers who apply high level of expert knowledge. Let us consider a kind of reverse method based on the following assumptions:

- finding the solution of the planning problem is very complex or not possible without proper model and solving engine
- solving problem of planning of production orders is achievable using feasible (in sense of feasibility of generated solutions and computational complexity) model of production planning problem
- construction of the model for chosen solving engine is complex task and using proposed approach is reasonable
- it is possible to estimate computational cost of solution search process for particular configuration of parameters value, describing the production situation
- we assume possibility of construction of sub problems model repository corresponding assumed class of problems occurring in considered class of production systems, we assume the dependencies involving elements of the models repository and mapping conditions of implementing environment can be represented as the formal knowledge base too
- particular production situation can be defined in form of declarative description by the assigned values form their domains.

4 Declarative description of production situation

The main elements determining sense of production planning and scheduling problems are: form of input data for planning procedure, kind of executed production processes, chosen optimization goals.

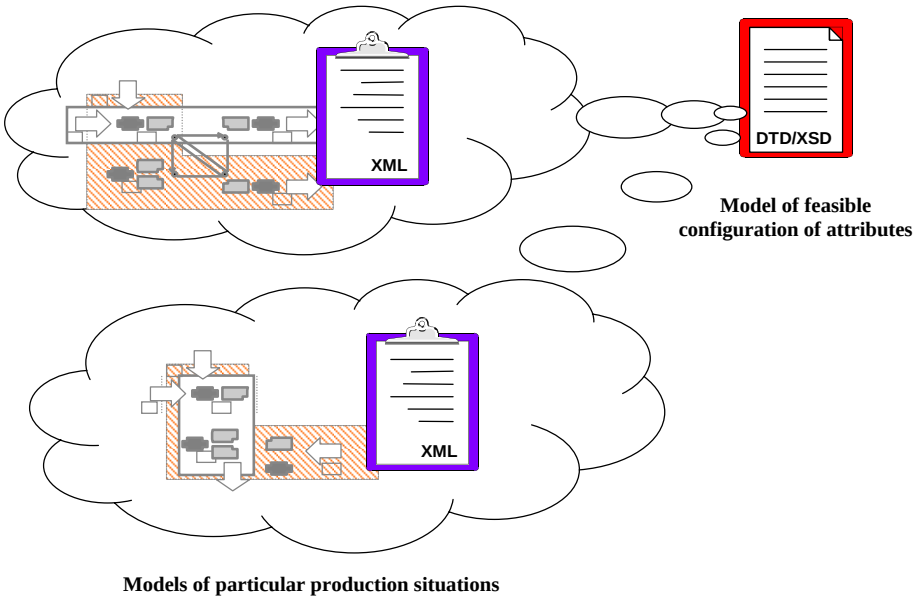


Fig. 3. Production situation description

In (Borowiecki, 2007) it is proposed to use XML as media to define production situation, as it is easy to define attributes and their domains using technology fundamentals. Formally production situation description is treated as an information table (Fig. 3). Data included in XML description of the production situation can be treated as information table PS and correspond to its attributes A. Values of the attributes are placed in contents of nodes and/or are represented by attributes value.

The DTD/XSD document guaranties the attributes form A of PS takes the values form their domains. So production situation description is $PS = (U_s, A)$ where U_s is not empty finite set of attributes value configurations (A is not empty finite set of attributes with defined finite domains). $\omega = \bigcup_{a \in A} \omega_a$ is information function and $f : U_s \times A \to \omega$. In (Borowiecki, 2007) it is assumed as in (Stefanowski, 2001), that the attributes are of different types: *nominal*, *serialized*, *scalar* or *criteria*.

5 Structure of planning procedure model

Unfortunately, application of the CP paradigm still causes many problems for software developers. Requirement of using completely separate paradigm with which software developers are not familiar, increases cost and the risk of such software projects realization. We suggest an alternative "interface to CP" to overcome this barrier.

Let us assume that in the research domain some class of production planning problems occurring in some class of production systems has been distinguished and mapped in form of CSP/COP scripts. We suggest to describe structure of (required in

particular production situation case) model of planning procedure in terms of their hierarchical structure and information flow (rather than strictly expose programmer to use in low level the CP solver semantics). In (Borowiecki, T., 2007) it is proposed an illustrated how to use appropriate definitions of partial sub models coded in form of XSD convention. Mentioned definitions describe partial models' (sub-problem models) elements such us: input parameters (corresponding to decision variables declaration, auxiliary data structures – declaration algorithms, constraints and distribution code sections (values assignment and search strategy to be possible to configure in particular sub problem model). An example repository of sub-problem models for an example class of production planning problems have been implemented in form of XML files due to XSD definition of the sub-problem model mentioned earlier.

Solving the particular sub problem represented by corresponding script formatted it the form of an XML file is possible due to defined code transformation process and using planning procedure model enabling code generated form particular XML description of elements of model repository to the CP solver environment.

Generally most of the practical production planning and scheduling problem (due to complexity in sense of computational cost and number of dependencies required to be mapped in resultant CSP/COP model code) consist of some hierarchical combined sub problem models. Independently mentioned structure (CP model code implementation of production planning procedure) can be obtained in traditional way or with suggested here approach (but without involvement of huge amount of expert knowledge) or proper support for prototyping process in our approach is essential.

An example.

Consider simple production planning problem. Let z_n is a set of work orders to be planned, $k(z_n)$ means part manufactured in work order z_n. $\nu(z_n)$ is the volume of the work order. The parts are manufactured due technological routes $r \in R_k$, the route is a set of activities j, executed on system resources $i \in I$. The task is to derive activities schedule assuring to fulfil requirements $\nu(z_n)$.

$$\sum_{r \in R(k)} \upsilon_r = \nu(z_n) \qquad \forall k(z_n) \qquad (1)$$

Decision process.

Solving the above formulated problem means to derive time moments t_j (starting time for operation j in route r. Taking in to account (1) the decision about number of parts υ_r manufactured due to r have straight influence to the number and type of operations to be scheduled. Assumption about unary resources (capacity of resources equals one) in production system structure requires to introduce constraints for t_j variables in form:

$t_{j2} - t_{j1} \geq p(j1,i)$ *xor* $t_{j1} - t_{j2} \geq p(j2,i)$

where $p(j1,i)$ means execution time of the activity j on the resource i (due to r).

Solving engine for well know scheduling problem with unary resources, especially using CSP/COP models and CP solvers can be provided as efficient even for large scale practical problems. But additionally considering the problem of choosing of production routes makes the task much more complicated (both form problem computational complexity and implementation manner).

Generally executable CSP/COP model of production planning problem can be treated as compound of hierarchical (procedural) implementation where superior sub problems are linked by declaration algorithms using some additional data structure allowing to execute solving process of interior sub problems (Fig. 4).

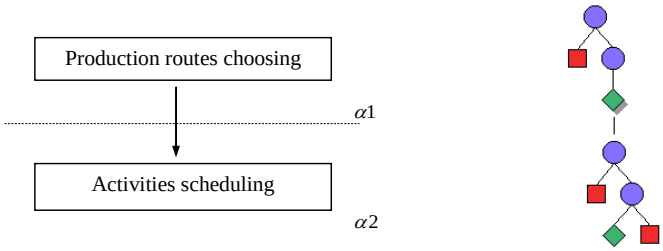


Fig. 4. Planning procedure model structure and corresponding search process

Due to declarative nature of CSP/COP problem models formulation we assume the solution of sub problem p allocated on α level in structure determines all values of $\mathbf{y}_p$ (and in general other elements of Y_p are accessible to -in particular optimal $\bar{\mathbf{y}}_p$ in sense of $w_p = \eta(x_p, \mathbf{y}_p)$). For the same reason declaration of all $\mathbf{x}_p$ input parameters have to be realizable not later than after search process of the last sub problem on level $\alpha - 1$ is accomplished.

In hierarchical structure of production planning procedure CSP/COP model we distinguish (due to implementation conditions):

- CSP/COP models of sub-problems as repository elements formatted in XML
- declaration algorithms, additional data structures, data structures implementing input and output sub problems parameters -coded as entity definitions

It is obvious that implementation conditions of chosen CP solver environment requires the code of particular sub problems models have to be properly combine with data structures, declarations algorithms and sub problems models code to be properly executed. So we have introduced two additional elements to the developed computational architecture: XSD definition for the structure of planning procedure model and XSLT transformation allowing to generate executable CSP/COP code.

6 Problem formulation

Two elements of developed computational architecture are illustrated in Fig. 5. First one as the manner allowing formally describe some class of production situation cases and the second aimed in support for prototyping of CSP/COP model for production planning procedure (without exposing software developer on low level CP solver code). As it can be noticed there is a gap between the mentioned elements. Basing on the elements of the approach illustrated above the problem of models structure identification for production orders execution can be formulated. We treat addressed identification problem as the configuration choosing problem. It is required to allocate chosen sub-problem models on the levels of hierarchical structure of prototyped model of production planning procedure due to: formal description of production situation (mentioned earlier), model of structure of production planning procedure and implementation manner of the modelling functional dependencies in sub models repository.

In (Borowiecki, 2007) it is illustrated the reasoning process based on the set of formally defined set of rules RU. Single rule ru takes the form of if-then if P than Q ($P \rightarrow Q$). P is a condition and Q is conclusion –notation for decision class K_n.

Rule's conditional part is conjunction of elementary conditions $cd = < cd_1 \wedge cd_2 \wedge ... \wedge cd_n >$. Elementary condition cd_i of rule ru is defined as: $(f(a_i, \omega) \propto term(a_i))$, where $f(a_i, \omega)$ is value of attribute a_i of ω; $\propto$ is elementary term for the relations of type $\{=, \neq, <, \leq, \geq, >, \in\}$ combining a_i attributes values from their domains. We assume the function $cd(a_i, \omega)$ returns values $\{0,1\}$. $(f(a_i, \omega) \propto term(a_i))$ is interpreted as fulfilment or not of the defined conditions. $\mathrm{K} = \{K_n\}$ states for set of decision classes. Assuming $d \notin A$, decision part (conclusion) for the rule can be represented as decision table $PSD = (\omega, A \cup \{d\})$ where d states for decision attribute $f(d, \omega) \in \mathrm{K}$.

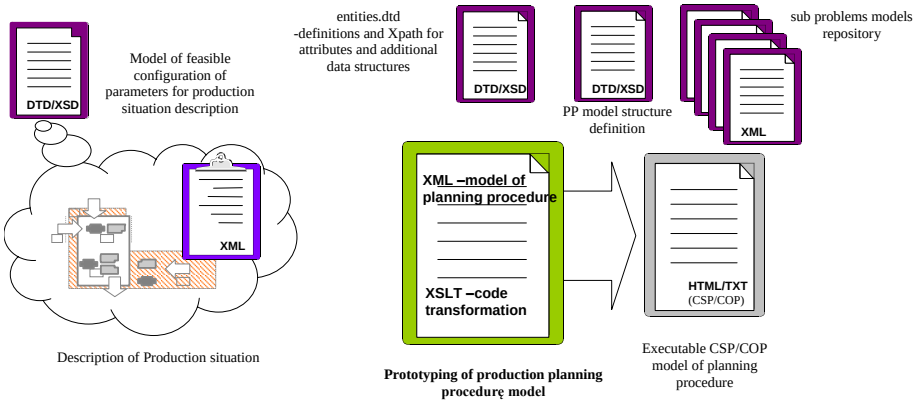


Fig. 5. Computational architecture for production situation description and production planning procedures models prototyping environment

Conclusions of the rules determine:

- allocation of the chosen sub problem models on levels of the structure of the script model of production planning procedure,
- which data structures are required, which are consist of determined values and which have to be derived due to propagation and search process.

Herein the reasoning process from (Borowiecki, 2007) is replaced by its CSP model representation. Attributes and their values of the XML description of production situation and XML implementation of the set of rules *RU* are transformed by the XML technological fundamental means. In our approach developed XSLT script generates executable CSP code of the problem of model's structure identification for production work orders execution problem.

In this chapter we address only two main elements of transformation process and resultant CSP/COP model. Let $Cds = \{cd_n\}$ be a set of all defined elementary conditions for rules RU. Considered model consists of binary variables cd_n corresponding to elements of Cds.

Conclusions for the rules rc_n correspond to pairs of finite domain variables: $(\{0,1\};\{0\#|\alpha|\})$. First element of the pair $(\{0,1\};\{0\#|\alpha|\})$ determines sub-model (and/or declaration algorithms, additional data structures). Second element of $(\{0,1\};\{0\#|\alpha|\})$ means level in prototyped model structure.

Model constraints.

The condition cd_n in transformation process is replaced by appropriate propagator. We consider herein the following types of rules: including, excluding, serializing, appointing determined data structures (no propagation and search process required).

Consider following set of decision classes $\mathrm{K} = \{\bar{p}_{\{n_i\}}, \tilde{p}_{\{n_i\}}, p\vec{r}_{\{n_i\}}, \bar{sd}, \tilde{sd}\}$ where: $\bar{p}$ means including rule, $\tilde{p}$ -consequently, excluding rule, $p\vec{r}$ - serializing rule, $\bar{sd}$ - making additional data structure as determined, $\tilde{sd}$ - rule marking data structure as forbidden (not accessible).

Rules activation means fulfilment of conjunction of their elementary conditions. From developed model point of view rule activation in transformation process introduces appropriate propagators narrowing domains of variables in pairs $(\{0,1\};\{0\#|\alpha|\})$.

E.g. including rule result (in transformation process) to propagator (we assume Mozart system as solving environment in this approach) the rule takes the form of {FD.sum [{FD.reified.sum *Dv* '<:' |*Dv*| } *Z*] '>:' 0}, serialization rules are transformed to pair of propagators {FD.impl {FD.reified.sum *Dv* '>:' |*Dv*|-1} and {FD.reified.sum [Z_1] '*a*' Z_2} 1}.

The second group of constraints is group related to dependencies between sub problem models. Formally, the relations between model structure elements in our

approach are represented in information table $IT = (U, V)$. V is the set of all data structures definitions being attributes of sub models and declaration algorithms. U - is admissible attributes configuration. V and end relations defines structure: $S = \prec V, \{R_m\} >$, where $R_m \subset \underbrace{V \times \ldots \times V}_{r(m)\ times}$ and most numerous relation.

In relation set R_m, $m = 1, \ldots, M$ we derive subsets (indistinguishable from modelling point of view) elements $\mathbf{v} \in R_m$. We annotate names for those elements, set of names is set of hyper edges E of the hyper graph of relations.

Consider Γ where, $< \mathbf{v}, e > \in \Gamma$ when e is the name of $\mathbf{v}$. $\Gamma = \bigcup_{m=1}^{M} \Gamma_m$, where $\Gamma_m = \{< \mathbf{v}, e > \in \Gamma : \mathbf{v} \in R_m\}$. Hypergraph representing dependencies between sub models is an triple: $H = \langle V, E, \Gamma \rangle$ where : V -set of vertex (data structures), E - set of names $\Gamma \subset \bigcup_{t=1}^{\infty} V^t \times E$, where V^t t - Cartesian product on set V.

7 Conclusions and future research

This chapter focuses on enabling capacity of Constraints Programming technology and tools, especially for general software developers interested in extending functionalities of ERP and MES systems, particularly in the area of production planning and scheduling.

Constraints Programming paradigm differs from general programing so most software developers are not familiar with. Traditional path concentring on steps such as requirements specification, problem identification, decomposition solving algorithms development, implementation seems to be high cost and risky due to requirements of high expert knowledge involved. Additionally differences in production systems organization requires form production planning software to be customizable sprightly to the needs (in traditional way of software development this means development of each project practically starts from the scratch).

The presented approach overcome both or those. Basing on parameters of production situation description, selection rules RU and dependencies model descripting repository elements using fundamentals of XML technology the computational architecture for automatically identification of models' structure of production planning procedures is illustrated. Practical verification (example implementation) of proposed approach has been made in (Borowiecki, 2007), such as example repository implementing typical production planning sub problems.

The whole implementation: elements of suggested architecture COP/CSP partial models and formulation of production planning procedures identification (omitting XML fundamental techniques and text editor module) uses Mozart system, but due the features of Constraints Programing paradigm it seems to be possible to change it by one of the popular Constraint Programing systems.

Growth of development tools, especially in area of DSL (Domain Specific Languages) and Software Incubation Environments allow to treat presented approach as a fundament usable with up to date programming techniques. The current work focuses on development of graphical tools (DSL editor based on Eclipse GMF) for models structures prototyping, on aspects of usability with new Constraints Programming solvers and on DSL tools for production situation description language.

References

Agrawal R., Kinzel G.L., Srinivasan R., & Ishii K. (1993). Engineering Constraint Management Based on an Occurrence Matrix Approach, *Journal of Mechanical Design*, 1993, Vol. 115.

Baptiste, P., Le Pape,C., & Nuijten, W., (2001). Constraint-Based Scheduling. Kluwer Academic Publishers.

Borowiecki, T., & Banaszak, Z. (2000). Production flow planning tasks integration with Constraints Programming paradigm. *Machine Tool Engineering* (in Polish), Wrocław, 2000.

Borowiecki, T., Politowicz, K., & Banaszak., Z.A. (2001). Customised Feature-Driven Computer Aided Production Planning Packages. A CLP Approach. *Proceedings of the 3rd Workshop on Constraint Programming for Decision and Control (CDPC'2001)*.

Borowiecki, T., Banaszak, Z., & Stryjski, R. (2004). An approach for CSP/COP modelling of decision making procedures, *Proceedings of the 6th Workshop on Constraint Programming for Decision and Control (CPDC'2004)*.

Borowiecki, T. (2007). Synthesis of Models of Production Planning Procedures for Work Orders Execution. PhD dissertation, Department of Electronics, Technical University of Poznań (in Polish).

Bradwell, R., Ford, J., Mills, P., Tsang, E., & Williams, R. (2000). An Overview of the CACP Project: Modelling and Solving Constraint Satisfaction/Optimisation Problems with Minimal Expert Intervention. *Constraints Programming 2000 Workshop on Analysis and Visualisation of Constraint Programs and Solvers*.

Fromherz, M.P.J. (2001). Constraint-based Scheduling. *Proceedings of the American Control Conference (AAC'01)*, Vol. 4: 3231-3244.

Gecode (2009). Generic Constraints development environment version 3.1. http://www.gcode.org/. Accessed 2014.X.12.

Hooker, J.N. (2004). A hybrid method for planning and scheduling. *Procedings of the 10th International Conference on Principles and Practice of Constraint Programming (CP'2004)*: 385-401. Springer.

Papalambros, P.Y., & Wilde, D.J. (2000). Principles of Optimal Design. Modelling and Computation, Cambridge University Press, New York, 2000.

Schulte, Ch., Smolka, G., & Wurtz, J. (1998). Finite Domain Constraint Programming. *DFKI OZ documentation series*, German Research Centre for Artificial Intelligence.

Stefanowski, J. (2001). Algorithms for induction of decision rules in knowledge discovery. DSc Dissertation (in Polish), Poznań University of Technology Press.

A Proposal of Decentralized Architecture for OKD-MES

Borja Ramis Ferrer

Tampere University of Technology, Tampere, Finland

borja.ramisferrer@tut.fi

Abstract. Recent research work in the industrial automation field determines that the computational power of embedded devices, which is used for process control on the shop floor, is sufficient for handling new functionalities. Therefore, it becomes possible to manage knowledge that is encapsulated in embedded devices, demonstrating a decentralized solution for controlling processes at the lowest level of the ISA-95 automation pyramid. This chapter argues that part of the OKD-MES functionality can be lowered to the device level. Moreover, the presented chapter exhibits that OKD-MES representation and management of knowledge can be distributed and handled in the shop floor, where devices are capable of controlling processes that are later executed by machines. Hence, this chapter offers an alternative for the actual architecture of OKD-MES, which is now centralized in terms of knowledge management. Furthermore, concepts, requirements and an early architecture for developing a decentralized OKD-MES are also shown and discussed.

1 Introduction and overview of the related research

This chapter presents an approach that lowers part of the OKD-MES functionality to the device level. The motivation of this research work is to propose an alternative for the OKD-MES centralized representation and management of knowledge. The presented approach describes how to achieve the decentralization of the entire system KB and the advantages that can offer to the actual OKD-MES.

The Manufacturing Execution System (MES) is a control system that is developed in the factory automation domain for monitoring and managing runtime processes on a shop floor. In the ISA-95 pyramidal structure (ISA-95), MES is situated between the Enterprise Resource Planning (ERP) and Supervisory Control and Data Acquisition (SCADA), filling the gap between the management and shop floor control. MES is integrated with the ERP for scheduling and dispatching orders to the control system, which increases the productivity by reducing manufacturing lead-time.

Furthermore, the on-going eScop project[1] is looking towards the use of ontologies and Service Oriented Architecture (SOA) for developing an alternative Open, Knowledge-Driven MES (OKD-MES). Through this platform, researchers plan to reduce costs and complexity of conventional MES. It can be stated that the three im-

[1] http://www.escop-project.eu/

portant players in the OKD-MES allowing researchers implement it and achieve the expected objectives: SOA, KR and embedded devices.

On one hand, SOA (Oracle, 2005) started to be used in the industrial automation domain for encapsulating functionality of system components and exposing it as services (Lobov et al., 2008). The software encapsulation is performed using the Web Service (WS) technology, which includes a set of standards to implement SOA. Some of the beneficial applications of WS are: cross-layer communication, integration of components that manages heterogeneous data and adaptation of existing applications (IBM, 2007). Moreover, SOA is used for monitoring and control of manufacturing system processes (Moctezuma et al., 2012).

Nevertheless, the degree of re-configurability and interoperability required for large-scale dynamic production systems are not covered entirely by the SOA paradigm. Fundamentally, the abstraction and storage of knowledge about the system status is required to allow the adaptation and re-configurability of systems. This is traduced in the implementation of Knowledge Representation (KR) in current manufacturing systems.

Through KR, engineers are able to describe physical and logical systems, allowing the collection of system data, which is later transformed in information that can be used by other subsystems. Examples of industrial automation research works that utilize a KB for storing data used for controlling processes are presented in (Lobov et al., 2009; Puttonen et al., 2013a; Ramis Ferrer et al., 2014).

As it can be seen in previously cited works, ontologies are used as a formal representation of manufacturing systems. Although there are many options for designing ontologies (Lastra et al., 2010) and a wide variety of languages (Maniraj & Sivakumar, 2010; Negri, et al., 2014), the common ontology language in this chapter is the Web Ontology Language (OWL) (OWL, 2004). Fundamentally, OWL has already been presented as a mature language for KR in factory automation (Lastra et al., 2006) and it has higher degree of representation than other ontology languages. In any case, OWL is a Resource Description Framework (RDF) (RDF, 2015), which is an Extensible Markup Language (XML) based language (Extensible Markup Language, XML). Consequently, the use of RDF-based languages for retrieving information of OWL KBs is appropriated. It should be noted that the refereed papers use SPARQL Protocol and RDF Query Language (SPARQL) (SPARQL, 2008) for querying ontological OWL models. Moreover, due to the adaptation and re-configurability of production lines, implementations must use another language, which permits the update of KBs. The SPARQL Update (SPARQL 1.1 Update, 2013) permits the modification of OWL models. An example on how SPAQRL and SPARQL Update can be utilized on industrial systems can be found in (Puttonen et al., 2013b). Hence, it can be stated that the implementation of KD approaches permits the runtime process control of systems in the industrial automation field. It should be noted then that the enormous advance that the use of ontologies offers in this domain is not only the knowledge representation, but also letting them play an important role in the system control.

Finally, as presented in (Iarovyi et al., 2013; Garetti et al., 2013; Moctezuma et al., 2012), Remote Terminal Units (RTUs) are a type of industrial controllers that are used in production lines for process control implementing SOA. More precisely, these

RTUs employ the Device Profile for Web Services (DPWS) technology for having the service functionality. Then, this DPWS devices are capable of real-time control, Web-based monitoring, and integration to SCADA systems, among other applications[2]. It should be noted that actual RTUs have increased their communication and networking capabilities, because the processing power of CPUs has also been increased. Thus, this chapter defends that due to the evolution of embedded devices the expanded resource power can be used for other functionalities, rather than just work as gateways for vertical communication in the automation pyramid. Among other functionalities, the possibility of hosting KR models is proposed.

Then, by combining SOA, KR and within the use of actual industrial controllers, the OKD-MES seems to be a promising concept for exploiting KD SOA implementations. On the other hand, the OKD-MES is currently being developed using KR as a centralized component for controlling processes (Fumagalli et al., 2014).

The remaining part of the chapter is structured as follows. Next section describes the actual OKD-MES architecture, explaining each layer's objective and functionality. Then third section presents a decentralized proposal for the OKD-MES, detailing how OKD-MES functionalities of knowledge management and representation can be reproduced and located in embedded devices. Finally, the fourth section concludes this chapter.

2 Present OKD-MES architecture

As it has been stated in previous section, the objective of the OKD-MES is to exploit the synergy of new generation of industrial controllers with a KD SOA for monitoring and controlling an entire automated manufacturing environment.

The paper (Garetti et al., 2013) presents an innovative solution for the control processes in manufacturing systems, based on integration of web-services technologies and an ontological model, allowing an easy configuration, update and scalability of the control system. In fact, this is the first presentation of the OKD-MES.

Furthermore, Garetti et al. (2013) present a comparison between the architecture of conventional MES and the OKD-MES. The main difference between the two architectures is the appearance of an ontology in the control layer. In addition, the control layer is divided in two layers for differentiate between the functionalities of representation and orchestration of WSs.

On the other hand, Fumagalli et al. (2014) presents the main concept of the eScop project, which is the combination of embedded systems and an ontology-driven service-based architecture for realizing the OKD-MES, defined also as a fully open automated manufacturing environment.

The architecture of the OKD-MES is presented in Fig. 1. It should be noted that Fig. 1 provides a detailed view of the eScop project kernel module presented in (Fumagalli et al., 2014).

[2] http://www.inicotech.com/

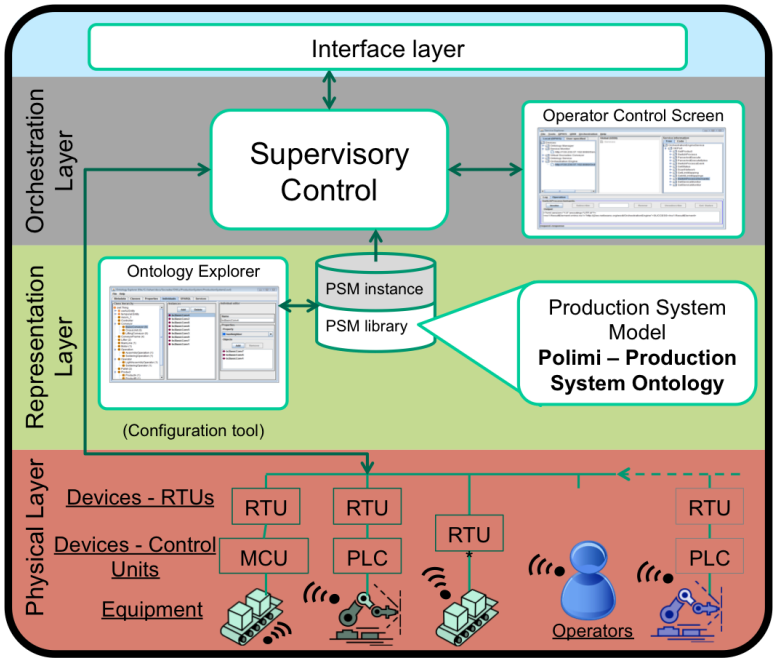


Fig. 1. Detailed view of the eScop project kernel (Fumagalli et al., 2014)

The architecture of the OKD-MES consists in five layers: Physical Layer (PHL), Representation Layer (RPL), Orchestration Layer (ORL), Interface (INT) and Visualization (VIS). It should be noted that VIS is not represented in Fig. 1.

The lowest layer is the PHL, which can be mapped to the to the hardware components located in the shop floor: machines, sensors, actuators, control units and the RTUs. The objective of PHL is the real-time control of devices. As it has been explained in previous section, these devices expose the functionalities of the production line as services and provide access to service description. In other words, RTUs serve as an interface between the PHL and other layers, permitting cross-layer communication.

Then, the RPL contains the ontological model of the production system domain. This model is known as the system KB. The Production Systems Model (PSM), which is based on the Politecnico di Milano Production Systems Ontology (P-PSO) (Garetti, 2012), is an example of model that can be included in the RPL. The formal representation of the production line status and service functionality supports the operation of SOA based orchestration tools, situated in the ORL.

The ORL hosts the Supervisory Control System (SCS) of the shop floor equipment. It should be noted that the SCS is composed by a set of orchestration tools and a production scheduler. Through these entities, the ORL is capable of composing and control the process that must be executed in the shop floor. The composition starts with incoming orders from the factory Order Entry System (OES), which indeed is exposed to the world via the INT.

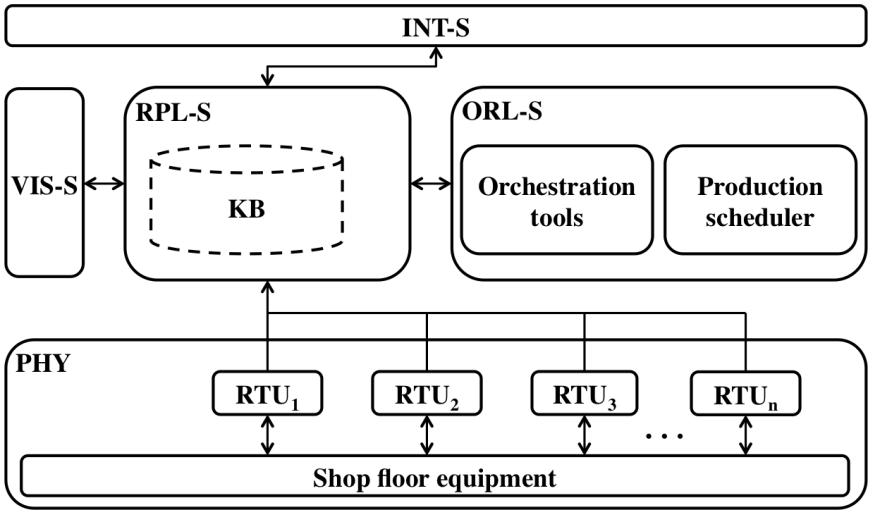


Fig. 2. Interaction of OKD-MES layers, implemented as services, with the RPL-S

In addition, the OKD-MES includes the VIS, which is as layer that is charge of offering graphical visualization of the OKD-MES actual status. Thus, this layer is used for runtime monitoring of the production system. It should be noted that the runtime visualization of processes is possible because this layer has an interface with the RPL, allowing the monitoring of the line status at any time.

Moreover, the research results presented in (Ramis et al., 2014) illustrate the first implementation prototype that follows the architecture presented in Fig. 1, however with two main differences: (1) the VIS and the INT layers are developed jointly in the User Interface service, and (2) the ORL is situated in the shop floor. In any case, the main concepts of the OKD-MES are tested, exposing layer functionalities as services. This is presented in (Ramis Ferrer et al., 2014) as the knowledge-based web service integration.

It should be noted that the RPL becomes a central component of the OKD-MES architecture because PHL, ORL, VIS and INT require the interaction with the model of the system. The eScop kernel representation exhibited in Fig. 1 is reshaped for easy understanding of the centralized architecture, which is exhibited in Fig. 2. Each diagram module of the presented OKD-MES that incorporates to the layer name the *"-S"* notation indicates that the functionality of the corresponding layer is implemented as a service.

The PHY functionality and integration with other layers is performed by the RTUs, which are used as the interface of the shop floor equipment and higher layers. Moreover, the number of RTUs situated in the PHY will vary according to the manufacturing system size and implementation, which is represented in Fig. 2 by three dots.

Finally, it should be noted that the concern of Fig. 2 is representation of the connections and components needed for representing the system knowledge and querying the KB for retrieving information or modifying the existing ontological model.

3 A Decentralized Architecture for OKD-MES

As it is explained in the first section, the processing power of CPUs at embedded devices, which serve as industrial controllers, has grown dramatically. This fact allowed the growth of their communication and networking capabilities. Due to the use of larger memories, RTUs have also increased the storage for resources as i.e. data and user programs. Then, the evolution caused that recent embedded devices have sufficient resources to perform deterministic control and networking functions.

On the other hand Fig. 2 exhibits that the actual OKD-MES architecture is centralized by means of interaction with the system KR. More precisely, the KB hosted in the RPL-S becomes a central component for the execution of system processes. This happens because any layer of the system must interact with the RPL-S.

Then, this chapter proposes the implementation of the RPL-S functionality inside the RTUs. The resulting embedded system is a device that not only keeps the objective of interfacing shop floor equipment with the rest of the MES, but also includes a similar instance of the RPL-S that is used for interaction with a KB. This device proposal is depicted in Fig. 3.

It should be noted that following considerations are applied to the device architecture presented in Fig. 3.

1. *"RP-S"* is a service, which is described and encapsulated in the embedded device, that implements the functionality of the OKD-MES RPL-S
2. The *"x"* notation is used for indicating a specific device that hosts its own KB. This means that i.e. $Device_1$ would host the KB_1

The main objective herein is to propose a decentralized architecture, as an alternative of the KR utilization in OKD-MES. For this purpose, each device hosts the description of (1) controlled equipment capabilities and (2) distributed network, in which all the devices are interconnected. This is traduced to an ontological model that is accessible by other devices and Layer-Services[3] within SPARQL and SPARQL Update endpoints. This is implemented as other embedded device interfaces.

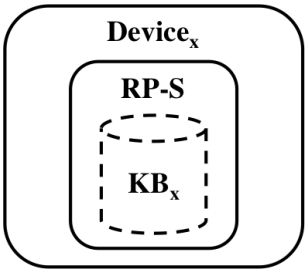


Fig. 3. Encapsulation of RPL-S functionalities in embedded devices

[3] Layer-Service refers to the service naming in actual OKD-MES architecture: VIS-S, INT-S, RPL-S or ORL-S

It can be stated that this approach aims at collaboration of intelligent embedded devices. The reason is because the presented approach designs a distributed network, in which devices are meant to become more autonomous for i.e. knowledge management and decision-making. Then, the implementation on new industrial controllers, which intends to work within the presented approach, must include the capability of:

- WS implementation
- Ontological model (e.g. OWL) hosting
- User program and apps implementation
- Interface for connection with shop floor equipment

By satisfying the first requirement of previous list, embedded devices are capable of implementing services for (1) controlling processes following cited SOA implementations in the industrial automation domain and (2) handling the functionality of the RP-S. Then, the ontological model hosting means that the device has sufficient resources for storing an OWL model. It should be noted that the model describes the required knowledge that the device needs for operating and performing its tasks. Moreover, the user programs and apps implementation is a requirement that allows designers to develop algorithms for i.e. knowledge management and device behavior. Finally, the interface for connection with shop floor equipment is needed for (1) being capable to activate inputs of machines, which are triggered from invoked operations by implemented WS and (2) being capable of reading system outputs for updating the actual system status, which is represented in the ontological model.

Once the device architecture is described, the disposition of them and interaction with remaining Layer-Services is depicted in Fig. 4.

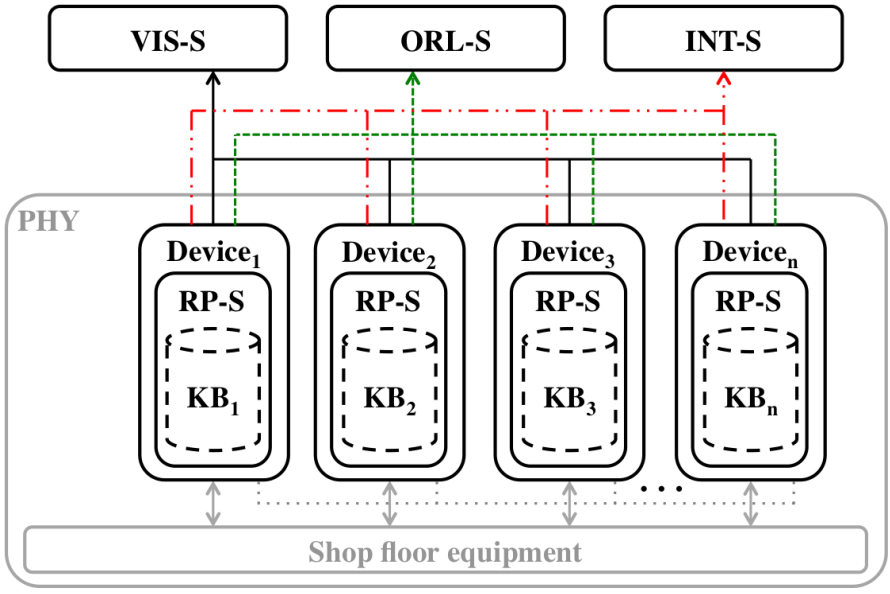


Fig. 4. Decentralized RPL-S alternative for the actual OKD-MES

As it can be seen, Fig. 4 exhibits that devices are connected to the shop floor equipment as it is depicted in Fig.2, but due to the encapsulation of a RP-S instance on each device, the represented knowledge is now decentralized. It should be noted that the union of all device KBs result in the entire manufacturing system model. Each device hosts the description on its own related functionality for avoiding redundant data in different embedded devices. Fundamentally, the duplication of data must be avoided because it occupies memory, which is resource power that can be used for other purposes i.e. for the increment of OWL data sets due to update queries.

Therefore, this approach imposes the need of managing the knowledge on device level. This management allows the response to incoming queries of other Layer-Service. For instance, if the ORL-S is invoking operations of certain process, it will need to request which device is in charge of the desired operation to be invoked. This request is delivered to the devices that, in cooperation, find a response and send it to the requestor. It should be noted that the management of KBs, which are encapsulated in embedded devices, in the factory automation domain, is described in (Ramis Ferrer et al., 2014), which presents a research work that is currently being reviewed to be published. In this paper, the algorithm and flow of queries within the distributed network of embedded devices is presented and demonstrated.

On the other hand, the encapsulation of KR in devices reduces efforts in horizontal communication and time of information exchange between RTUs, in comparison to the recent OKD-MES, devices and the RPL-S. The reason is that the encapsulation of the RP-S in devices allows the direct interconnection between shop floor equipment and its model, without intermediate layers.

The rest of the interactions and functionalities remain in a similar distribution to the recent OKD-MES architecture. Thus, the INT-S is a service in charge of exposing the functionalities of OKD-MES to the outside of the system. On the other hand, the VIS-S is used for monitoring of shop floor processes i.e. as an HMI. Finally, the ORL-S interacts with devices for executing operations, which are described in the local resources of the industrial controllers. It should be noted that the interconnection of each Laser-Service with devices is implemented as a bus or communication channel, which has implemented its corresponding interface in the RP-S instance. In this manner, devices handle requests from any Layer-Service independently of its type. More precisely, cause the ontological model is described using OWL, queries from any Layer-Service are sent to the device within SPARQL over HTTP endpoint, which is described in (SPARQL 1.1 Graph Store HTTP Protocol, 2013). This interface is a bidirectional interconnection that devices use also for sending responses to Layer-Service requests.

4 Conclusions and Further Work

Firstly, this chapter presents the actual layer organization and functionality of the recent OKD-MES. Afterwards, proposes a decentralized architecture as an alternative for lowering the management of knowledge to the device level. In fact, it is concluded that this new approach offers a reduction of vertical communication since part of the

interactions between the PHY and RPL is replaced by horizontal communication. In other words, as it has been presented in Section 3, some of the required OKD-MES layer-to-layer communication is replaced by a device-to-device interaction, due to the encapsulation of functionality at device level.

Likewise to the encapsulation of the RPL-S OKD-MES functionalities inside embedded devices, this approach could make an ambitious further step and also propose the encapsulation of other Layer-Service functionalities. This progressive approach would offer same kind of benefits as i.e. avoiding vertical communication, which reduces efforts of implementation and time in the information exchange, with other layers of the OKD-MES.

However, the limits of encapsulation in RTUs and required algorithms for device cooperation to substitute other layer functionalities must be still studied. Hence, the further work on presented approach is to research the limits and the methodology of continuously lowering functionalities to the device level, where the processes are finally executed. Potentially, the next step of research work is to implement devices that aside from RP-S functionalities, are able also host as local resources the visualization of themselves, which i.e. can feed the information required of a dedicated HMI.

References

Extensible Markup Language (XML). http://www.w3.org/XML/. Accessed 2015.04.01.

Fumagalli, L., Pala, S., Garetti, M., & Negri, E. (2014). Ontology-based modeling of manufacturing and logistics systems for a new MES architecture, *IFIP Advances in Information and Communication Technology*, Vol. 438: 192-200.

Garetti, M., (2012). P-PSO Ontology for Manufacturing Systems. *Information Control Problems in Manufacturing, Vol.14/1*: 449–456. http://doi.org/10.3182/20120523-3-RO-2023.00222. Accessed 2015.IV.01.

Garetti M., Fumagalli L., Lobov A., Martinez Lastra J. L. (2013). Open automation of manufacturing systems through integration of ontology and web services. *Proceedings of 7th IFAC Conference on Manufacturing Modelling, Management, and Control*, Saint Petersburg, Russia, June 19-21, 2013.

Iarovyi, S., Garcia, J., & Lastra, J. L. M. (2013). An approach for OSGi and DPWS interoperability: Bridging enterprise application with shop-floor. Proceedings of the *11th IEEE International Conference on Industrial Informatics (INDIN'2013)*: 200-205.

IBM (2007). IBM developerWorks: New to SOA and web services. http://www.ibm.com/developerworks/webservices/newto/service.html. Accessed 2015.04.01.

ISA-95. ISA-95 standard for the integration of enterprise and control systems. http://www.isa-95.com/subpages/home/welcome.php. Accessed 2015.III.05.

Lastra, J. L. M., & Delamer, I. M. (2006). Semantic web services in factory automation: fundamental insights and research roadmap. *IEEE Transactions on Industrial Informatics*, 2(1): 1–11. http://doi.org/10.1109/TII.2005.862144. Accessed 2015.IV.01.

Lastra, J. L. M., Delamer, I. M., & Ubis, F. (2010). Domain Ontologies for Reasoning Machines in Factory Automation. ISA.

Lobov, A., Puttonen, J., Herrera, V. V., Andiappan, R., & Lastra, J. L. M. (2008). Service oriented architecture in developing of loosely-coupled manufacturing systems. P*roceedings of the 6th IEEE International Conference on Industrial In formatics (INDIN'2008)*: 791–796.

Lobov, A., Lopez, F. U., Herrera, V. V., Puttonen, J., & Lastra, J. L. M. (2009). Semantic Web Services framework for manufacturing industries. *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO'2008)*: 2104–2108.

Maniraj, V., & Sivakumar, R. (2010). Ontology Languages–A Review. *International Journal of Computer Theory and Engineering*, Vol. 2/6: 887–891.

Moctezuma, L. E. G., Jokinen, J., Postelnicu, C., & Lastra, J. L. M. (2012). Retrofitting a factory automation system to address market needs and societal changes. *Proceedings of the 10th IEEE International Conference on Industrial Informatics (INDIN'2012)*: 413–418.

Negri, E., Fumagalli L., Garetti, M., Tanca, L. (2014). A review of semantic languages for the conceptual modelling of the manufacturing domain. *Materials of XIX Summer School "F. Turco" A Challenge for the future: the role of industrial engineering in a global sustainable economy*. Senigallia, Italy, September 9-12, 2014.

Oracle (2005). SOA and Web Services. http://www.oracle.com/technetwork/articles/javase/soa-142870.html. Accessed 2015.IV.01

OWL (2004). Web Ontology Language Reference. http://www.w3.org/TR/owl-ref/. Accessed 2015.IV.01.

Puttonen, J., Lobov, A., & Lastra, J. L. M. (2013a). Maintaining a Dynamic View of Semantic Web Services Representing Factory Automation Systems, *IEEE 20th International Conference on Web Services (ICWS)*: 419–426. http://doi.org/10.1109/ICWS.2013.63. Accessed 2015.IV.01.

Puttonen, J., Lobov, A., & Lastra, J. L. M. (2013b). Semantics-Based Composition of Factory Automation Processes Encapsulated by Web Services, *IEEE Transactions on Industrial Informatics*, 9(4), 2349–2359. http://doi.org/10.1109/TII.2012.2220554. Accessed 2015.IV.01.

Ramis Ferrer, B. (2013). An ontological approach for modelling configuration of factory-wide data integration systems based on IEC-61499. Tampere University of Technology, Finland. https://dspace.cc.tut.fi/dpub/handle/123456789/21541. Accessed 2015.IV.01.

Ramis Ferrer, B., Gonzalez, L., Iarovyi, S., Lobov, A., Lastra, J. L.M., Vyatkin, V., & Dai, W. (2014). Knowledge-based web service integration for industrial automation, *12th IEEE International Conference on Industrial Informatics (INDIN)*: 733–739. http://doi.org/10.1109/INDIN.2014.6945604. Accessed 2015.IV.01.

RDF (2015). Semantic Web Standards. http://www.w3.org/RDF/. Accessed 2015.IV.01.

SPARQL (2008). SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/. Accessed 2015.IV.01.

SPARQL 1.1 Update (2013). http://www.w3.org/TR/sparql11-update/. Accessed 2015.IV.01.

SPARQL 1.1 Graph Store HTTP Protocol (2013). http://www.w3.org/TR/sparql11-http-rdf-update/. Accessed 2015.IV.01.

REST-Enabled Physical Devices in Service Oriented Architecture

Ondřej Severa and Roman Pišl

University of West Bohemia, Plzeň, Czech Republic

{osevera, rpisl}@ntis.zcu.cz

Abstract. Current developments in the field of embedded devices create new opportunities for the researchers and developers. Increased computational power and memory size, and lowered consumption of power allow developers to create and use high level function built on top of more powerful frameworks. It also brings a space for the Service-Oriented Architecture (SOA), where multiple devices with common interface can be used for solving difficult tasks. This chapter describes specification of an interface for physical devices (PLC, RTU) which incorporates REST API. Physical device is directly connected to the physical part of the controlled process. Depending on the type, it contains direct inputs and outputs or advances web-service management. Such a device can be controlled (orchestrated) with simple HTTP requests.

1 Introduction

Nowadays embedded devices are more and more important. Increased computational power and memory size and overall performance enable extended scope of applications. The new direction commonly called the Internet of Things (IoT), which together with home automation, tends to be one of the targets for embedded platforms. This brings new demands for interconnection of these devices. Usage of service oriented architecture (SOA) in the embedded devices unifies interfaces and eases the communication. Legacy SOA was based on the SOAP and WSDL web services, but this solution was very heavy. The lightweight communication is requested. This can be provided via REST (Fielding, 2000) which is not a standard, but a set of recommendations on using well known and tested HTTP protocol.

One of the important things for the embedded device is discovery. One wants to know how many devices are accessible thought the network, what their addresses are and what are their capabilities. There are several solutions except of WSDL with WS-Discovery such as Universal Plug and Play (UPnP). Discovery through UPnP combine with REST is researched in (Fernandez Alcal et al. 2010; Parra et al., 2014; Newmarch, 2005). Several frameworks based on the web services are designed in (Corujo et al. 2005; Kopecky et al. 2008). The REST protocol is extended to be more fault-tolerant in (Maciel de Silva and Hirata, 2013). Also the lightweight application, e.g. (Cheng et al., 2015), points that REST is gaining place in embedded devices.

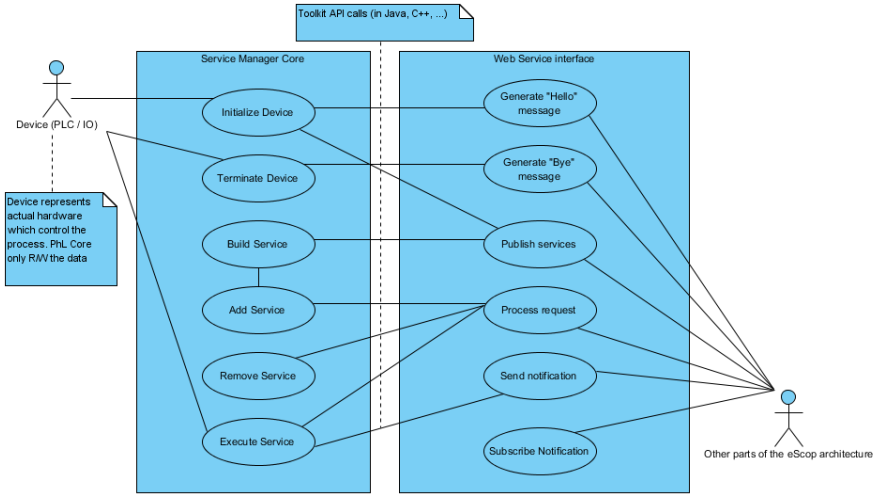


Fig. 1. Components and use cases for Service Manager

This chapter presents several classes of REST enabled physical devices connected directly to the physical part of the controlled process. The need for such a division and specification flows from the solving the eScop project (Embedded systems Service-based Control for Open manufacturing and Process automation) especially in design of the Physical layer (the last interface between controlled process run in the embedded device such a Raspberry Pi). The Physical layer is composed from a number of devices each containing the Web Service Interface and the Service Manager.

The Service Manager represents the interface layer between simple input/output device in the field (PLC, RTU etc.) and the remaining part of the eScop System. A general view on the Service Manager architecture is shown in Fig. 1. The Service Manager is divided into two main parts: Service Manager Core and Web Service Interface. These two components are described in detail in the next sections.

This simple separation of Service Manager into two functional components should help understanding principles of Service Manager application and should also ease the creation of a functional specification.

The main purpose of the Web Service Interface is publishing the whole device functionality to the eScop System by Web Services. Principles of this component are described in section 3.

The Service Manager Core is responsible for communication with simple input/output devices in the field and all the algorithms that are connected with the handling of services. Principles of this component are described in the section 4.

Presented specification for Web service interface will be tested on top of the INCAS mini-pilot simulator described in another chapter of this book (Balda and Štetina). The simulation uses REX Control System running on a standard device.

1.1 Service Manager Core

The Service Manager Core component (Core) is responsible for communication with simple input/output devices in the field and all the algorithms that are connected with the handling of services. This component integrates the whole logic that is connected with:

- initializing services,
- adding and building services,
- executing services,
- terminating services.

It also implements all the protocols that are needed for communication with devices in the field. To publish its functionality, the Core uses the Web Service Interface component.

1.2 Web Service Interface

The Web Service Interface is a component of Service Manager that is responsible for publishing its functionality using Web Services. Possibly it can be a separate application or its part (in-process component). For the eScop System, it must be capable to provide REST interface on the public side and provide specific API for the Core component (private side). It should provide the following set of functions:

- Processing of the discovery request,
- Code generator management,
- Notification management,
- Request management.

2 eScop device

This section identifies all the types of devices used in the Physical Layer of the eScop System and describes their functionality. Generally, a device, which conforms to the functionality described in this section, can be labeled as an eScop Device of the Physical Layer. All eScop Devices have several common attributes. These common attributes of eScop devices are:

- they have a public REST interface,
- the device is discoverable and identifiable using standard eScop mechanism.

The three main types of devices are:

- eScop Simple I/O,
- eScop Service Manager,
- eScop RTU.

The functionality of these types of device is described in the following subsections.

2.1 eScop Simple I/O

The eScop Simple I/O device depicted in Fig. 2 is a basic device in the Physical Layer. Although its whole functionality is accessible through Web Services, it provides only basic interface for reading and writing data values. It is expected that in most cases, these data values will be direct values of the device's inputs and outputs, so called raw data. The only limitation of eScop Simple I/O device lies in the REST interface it provides, so that only data values can be read or written. The specification of the REST interface of the Simple I/O can be found in section 3.3.

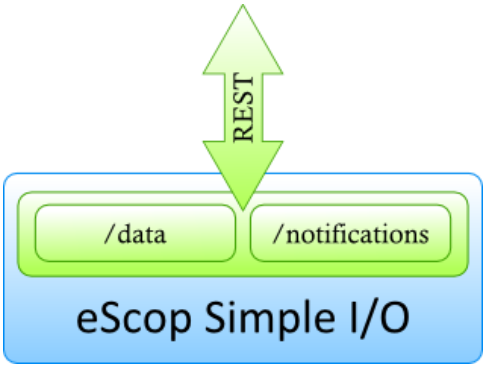


Fig. 2. REST Simple I/O device

2.2 eScop Service Manager

The eScop Service Manager device shown in Fig. 3 is a more advanced device on the Physical Layer that is capable of running more complex operations (Services). The main difference from the Simple I/O device is the possibility of creating, modifying and updating Services in the device at runtime. The specification of the REST interface of the eScop Service Manager can be found in the section 3.3.

Fig. 3. REST Service Manager device

2.3 eScop RTU

The eScop RTU device depicted in Fig. 4 is the most complex and complete device on the Physical Layer. It is a combination of the Simple I/O device and Service Manager. That means anyone can connect to the RTU device using standardized REST interface described in this chapter and can control such device either by reading or writing data values or by invoking Services. Eventually, these services can be created, modified, updated or terminated upon request at runtime. It is expected, that an eScop RTU should be distributed with some basic functionality that includes an I/O interface, internal logic and some set of predefined Services. More complex Services can be later defined and implemented using the Service Manager functionality of the device. These Services may be created by skilled user or programmatically from upper layers of the eScop System; for example from the Representational Layer (eScop ontology). Possible basic implementation architecture of RTU is depicted in Fig. 4.

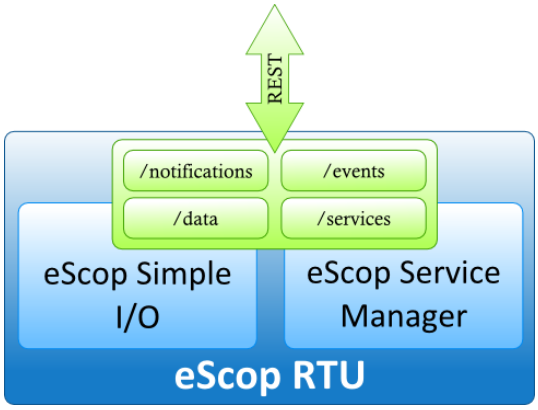


Fig. 4. eScop RTU

The identification of the Simple I/O component and the Service Manager component in the eScop RTU device also allows reusing existing devices in the field (PLC, controllers etc.) as a basis for eScop RTU devices. For example, for existing PLC a REST interface can be implemented, so that an eScop Simple I/O device is created. When a Service Manager component is implemented and added to that device, an eScop RTU is created.

It is expected, that such a solution with existing PLC that will be extended to conform eScop RTU specification will be used in an eScop pilot application. The specification of Physical Layer of eScop project and the Pilot Application placed some strong requirements on eScop RTU functionality that are not within the scope of the eScop project. It would be also very inefficient to "reinvent the wheel" by re-implementing PLC functionality that is already available in the field. Instead of reinventing such a functionality, the eScop solution of Physical Layer will focus on identifying existing devices in the field, that can be used for eScop System and extend functionality of these devices by implementing and adding components that are described in this chapter.

3 Web Service Interface

This section provides a basic introduction into architectures and technologies being used, which concern Web Services for eScop System and specification of a Web Service Interface (Interface) of eScop Devices and Service Manager. Any implementation of eScop Web Service Interface should follow this specification.

3.1 Web Services

There are several ways to create Web Services for service oriented architectures. One of the combinations is SOAP with WSDL description. This is a mature solution but very heavy. In the current trend there is a need to create the Web Service interface as simple as possible it is because of the REST - Representational state transfer which is used. The next section will describe this principle and proposes the description language of such service.

REST - Representational State Transfer.

REST (Fielding, 2000) defines a set of architectural principles by which one can design Web services that focus on a system's resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages. If measured by the number of Web services that use it, REST has emerged in the last few years alone as a predominant Web service design model. In fact, REST has had such a large impact on the Web that it has mostly displaced SOAP - and WSDL-based interface design because it is a considerably simpler style to use.

Nowadays a lot of web leading companies are converting their Web Service APIs to the purest form of REST where a concrete implementation of a REST Web service follows four basic design principles:

- Use HTTP methods explicitly.
- Be stateless.
- Expose directory structure-like URIs.
- Transfer JavaScript Object Notation (JSON), XML, or both.

Swagger.

Swagger (Reverb Technologies, 2014) is a specification and complete framework implementation for describing, producing, consuming, and visualizing RESTful web services. It is language-agnostic specification. Swagger is open-source project developed by Reverb Company. It is released under the Apache License, Version 2.0. Swagger describes REST API using JSON text file format. The specification contains set of rules how final description file should look like but the final look and function of the API is left for the developer.

Swagger specification is suitable for describing the services in eScop device because the description is in JSON format and can be read both by humans and computers. It is language-agnostic so it is not directly bound to some program language.

Since it is only description tool there are no restrictions on the final look of the API. Last but not least, it is an open-source project with large active community and many applications in various Web Services.

3.2 General Principles

There are several common principles for eScop REST interface. If followed, one should be able connect and read common information from the entire eScop System regardless of the layer (Physical, Orchestration or Representation). This section describes some general principles and division of the REST interface. Section 3.3 contains proposed specification for each of the components.

Data.
Data interface represents raw data from the device (RTU, PLC, ...) or application (RPL, ORL). The interface is the bridge between the application and the rest of the eScop architecture. Data can be simply read or written, but they should not be representing functions. Each item is unambiguously identified by URL (e.g. http://deviceURL/data/item1). There is no need for any parameter in the request for reading the data value. Write data value request contains only the new value which will be updated. This interface represents the fastest and easiest way to read or write data to the eScop device. It is primary synchronous, because raw data should be available at any time. Any other complex behavior should be implemented as a service (see the Service specification below).

Notifications.
Notification represents implementation of the publish/subscribe mechanism. When a user wants to be notified on data change/update or periodically one can add a new notification to the notification server. The notification server can be part of the eScop device (i.e. eScop Simple I/O) or work as a standalone application. It uses REST Web Services and maps Create, Read, Update and Delete (CRUD) functions to HTTP verbs POST, GET, PUT and DELETE. Each notification is represented as a REST resource so one can list, subscribe notifications, create new ones and also remove them simply using the HTTP methods.

The data resource (producer) is specified by the URL. It can be any resource where one can call HTTP GET method (e.g. http://deviceURL/data/item1). When the Client (consumer) subscribes for the notification it has to provide destination resource also described by URL (e.g. http://clientURL/data/updatableItem) which can be updated by a PUT or POST HTTP request.

Events.
Events differ from data or notifications. They represent some discrete events which happen during service or execution of the application. The user cannot add a new event unless he defines it inside the application or service configuration. The user is

only able to subscribe and unsubscribe for the event (add or remove new on change notification). Unlike the Data interface it is not possible to read event data when no event is emitted, one can only read the description and the event response format. Each event must be described by a unique URL.

Services.
The interface for services contains a list of all functions / operations which the user (or other part of the eScop System) can invoke. Services are application and configuration specific, each layer supports different set of services. A service can be called both synchronously and/or asynchronously; it can trigger several events during its execution. In the Service Manager core, one can freely add or remove services; the mechanism is explained in the following section.

3.3 Specification

This specification describes the RESTful interface of any eScop compliant device. All parts are optional, but when implemented, they must follow this specification. The specification is described using Swagger. It is stored in JSON format on the server and is freely accessible for any other eScop device or directly to user or developer.

Each device has a unique root URL. Every other resource can be defined by absolute or relative path to this URL. Any device can be browsed as a tree. There are two node types, simple leaf and folder node. Each folder node contains at least the same data as NodeInfo class (Table 1).

Table 1. NodeInfo class description

Properties	*Description*
name	Name of the node
links	An object with links, at least with self URL, which points to this resource
isFolder	Boolean flag if the node has children
children	An array of child nodes, which should contain at least the same information as NodeInfo

The previous section describes division of eScop devices to several types: Simple I/O, Service Manager, eScop RTU. Each of these devices supports different interfaces except eScop RTU, which implements all of them. Fig. 4 shows that Simple I/O implements Data and Notification interface and Service Manager contain Events and Service interface. Manager must also implement Notification interface because it is a general mechanism how to subscribe for data changes. This chapter describes the basic idea for specification of each interface, but the whole specification with descriptions is generated from Swagger documentation tools. The API description will be available in any device through /api/api-docs URL. One can use swagger-ui tool (/api/api-viewer) to display current version of the API for the device. From above flows we can find default structure for the eScop devices (Table 2).

Table 2. Default structure of REST interface API for various eScop devices

Simple I/O	Service Manager	eScop RTU
/data	/notifications	/data
/notifications	/events	/notifications
/api/api-docs	/services	/events
/api/api-viewer (optional)	/api/api-docs	/services
	/api/api-viewer	/api/api-docs
		/api/api-viewer

Data.
The Data interface should be the fastest. The user cannot add or remove resources; they are defined by underlying structure of PLC, layer application etc. There are two functions, which can be invoked: READ and WRITE. They can be called by HTTP GET or PUT method. Each leaf node returns simple data object (Table 3).

Table 3. DataValue

Properties	*Description*
v	Value where type of the value is specified in DataInfo object
q	Quality (BAD, UNCERTAIN, GOOD)
t	Timestamp in milliseconds from UNIX Epoch

There are general principles for data:

- Every leaf node in data can be nested in tree structure, where each parent has at least the same property as NodeInfo (e.g. /data/system1/item1).
- Every leaf node supports /info resource, which returns DataInfo object (structure which describes type of the value, its mode).
- Every leaf node supports /notification resource which list current subscribe notification for this data resource. The user can also add or remove the notifications. It is the shortcut where one does not have to insert sourceURL in subscription request. (See next section for more details).

Notifications.
Notifications resource contains a list of all subscribed notification for the device. The user can list, add new or remove any selected notification. Each notification object contains at least the properties listed in Table 4.

Events.
Events are similar to the data; one can browse them, read the event info and subscribe for the notification. The main difference between data and events was described in previous section. The data from an event are not accessible unless the event is emitted. In section 4 is described how the events are introduced to the system. Every event should comply with the specification given in Table 5.

Table 4. Notification

Properties	*Description*
sourceUrl	URL of local resource. It can point to any value that one can call GET function. The URL can be in 'full' format including the server address (e.g. 'http://server:port/data/point1') or in 'local' format ('/data/point1')
destUrl	URL of remote resource. It can point to any value that one can call PUT or POST function. URL MUST be in 'full' format including the server address (e.g. 'http://destServer:port/services/notify')
minUpdatePeriod	Minimal update period in seconds. Default: 1 second.
maxUpdatePeriod	Maximal update period in seconds. Default: 1 minute.
type	Periodical will send the data every minUpdatePeriod. Change will send the data whenever any local resource changes.
method	HTML method used during notification call.
clientData	Arbitrary client object serialize to String representation. This object will be attached to every notification message

Table 5. Event

Properties	*Description*
name	Case sensitive name of the event. It MUST be unique
lastEmit	Timestamp in milliseconds from UNIX Epoch
data	Each event can have different format of the data or follow some general scheme

Services.
As was mentioned in previous section, Services represents functions or operations provided by the particular device. Each service is described using the /info resource. This resource returns list of all input and output parameters together with description of the service function. The important thing is that /info for service contains a semantic description of the service in eScop service semantic description format (Currently considered to be OWL-S).

Provided function depends on the current configuration of the device. The user is able to add or remove services by providing their description in Structured text (see section 4.2 for more details). Response for /info request returns the description at least in NodeInfo format.

4 Service Manager Core

This section contains the specification of the Service Manager Core (Core), the core software component of the eScop Service Manager and eScop RTU. Any implementation of Service Manager Core or eScop Service Manager should follow this specification.

4.1 Main functions

A general view on the Service Manager architecture was shown in Fig. 1. It is expected, that the following basic functions will be provided by the Service Manager Core:

Initialize Device.
This function initializes the whole device and makes it ready for operation. All other functions called using Web Service Interface must be done after initialization. It is expected, that such call will finish with error.

The "Initialize" function should be called internally by a device itself upon startup. After successful initialization of the device, all subsequent calls of initialization have no effects and should return successful error code. When initialization ends with error, the device should be recognized as uninitialized and subsequent calls to "Initialize" function should start complete initialization from the beginning. Error code returned by the function should carry information about the reason of the failure.

Terminate Device.
This function safely cancels all running tasks in the device and removes the device from operation. No other function should be called using Web Service Interface after device termination. It is expected, that such call will finish with error. There are several ways how to initiate the termination of the device:

- issuing a termination command through Web Service Interface,
- issuing power-off trigger by pushing power-off button of the device or by software,
- issuing a critical software or hardware fault.

Add Service.
This function allows to dynamically adding a Service into the device. The format and structure of dynamically added service is described in section 4.2.

Build Service.
When adding a Service using the REST interface, the service is described in a human readable format that is described in section 3.3. Before the Service can be turned into operation, it has to be transformed and compiled into internal binary representation that the Service Manager is able to interpret. This operation is handled by the "Build Service" function. After the Service is successfully added and built, it is immediately accessible through the REST interface.

Execute Service.
This function executes a Service from its internal binary or text representation. After successful registration, the Service may be executed. There are two modes of execution of a Service:

1. execution upon request,
2. periodical execution.

Execution upon request is initiated from the Interface. Any client connected through the Web Service Interface can execute an arbitrary Service that is capable of being executed upon request. This mode of execution represents the standard Web Service. Periodical execution of a Service is initiated periodically from the Core. There is no client needed, that has to invoke the Service through the Interface. The main purpose of Service executed upon request is the implementation of standard Web Services. The main purpose of Services executed periodically is to manage periodical tasks that have to be executed above the PLC or Simple I/O device.

The difference between these two types of Services also lies in the way the Service is communicating data to the caller. Since there is no caller of a periodical Service (the real caller is the Core), the Service doesn't return any data. However, the Service can trigger an Event which will lead to generation of Notification that notifies all registered Clients about the Event.

Both types of Services are identical in other aspects. Any Service can read arbitrary data from underlying PLC or Simple I/O device and write data back. During its execution, the Service can invoke Events. Clients, which have registered Notification on the Event, are immediately notified about the Event through Interface.

Remove Service.

This function terminates a Service when it is running, closes its REST interface, deregisters the Service from the Service manager and processes all cleanups needed. After the call of this function, the Service is no longer available in the device and all calls to the service through the REST interface leads to error.

Generating services using structured text.

For programming of Devices, the Structured text from the standard IEC 61131-3 was chosen as the main programming language. It was decided at this point, that Structured text should be the main language for representation of Services.

Each Service defined through Interface should be provided in Structured text (STL) by the client. Each Service may be defined in a separate single text file or object. It should be also possible to include several Service or Event definitions into one text file or object. An example of a basic Service defined by STL is shown in the following preliminary code snippets.

```
Definition of the global variable
VAR
 status : INT;
END_VAR;
Definition of the event
{funcmode=event, description='Buffer is full'}
FUNCTION EventFull ;
  VAR_INPUT
    {description='Name of the conveyor'}
```

```
   conveyor : STRING[20]{};
  END_VAR;
END_FUNCTION;
```
Definition of the periodic function
```
{funcmode=periodic, period=100ms}
FUNCTION task ;
  VAR
    tmp : INT;
  END_VAR;
  tmp := GetValue('simulation/T1/FULL');
  IF (tmp > 0) and (status <10) THEN
      status :=10;
      EventFull('T1');
  END_IF;
END_FUNCTION;
```
Definition of the service
```
{funcmode=service, description='TransferIn Web Service'}
FUNCTION TransferIn : INT ;
  VAR_INPUT
    {description='Name of the conveyor'}
    conveyor : STRING[20];
  END_VAR;
  VAR_OUTPUT
    {description='Number of boxes in the queue'}
    count : INT;
  END_VAR;
  VAR
    tmp : INT;
  END_VAR;
  tmp := GetValue('simulation/'+conveyor+'/FULL');
  IF tmp > 0 THEN
    TransferIn := -100;
    RETURN;
  END_IF;
  count := GetValue('simulation/'+conveyor+'/count');
  SetValue('simulation/'+conveyor+'/MP_START/BSTATE',TRUE);
  TransferIn := 0;
  RETURN;
END_FUNCTION;
```

The first function, EventFull, has no body definition, since it is only an Event prototype. According to this prototype, an Event object will be created and published through the Interface. This Event takes one parameter of the string type, which identifies the name of a conveyor that originates the Event.

The second function, task, is simple periodic Service (can also be called “Task”), that is periodically executed on the Service Manager every 100ms. It periodically watches the state of the variable *simulation/T1/FULL* and invokes an Event when the variable is non-zero and the state is not full.

The third function, TransferIn, is a standard Web Service. It takes one parameter of the string type that identifies the name of a conveyor that should be transferred and returns one parameter that indicates the number of boxes loaded on the conveyor.

This chapter presents only this example as a simple illustration of the basic principles of Service code generation. The way metadata (inputs, outputs) will be automatically extracted from STL and published through Interface, the principles of invoking Events and reading and writing data from/to the underlying PLC or Simple I/O Device will be specified in details in future publications.

5 Conclusions

The purpose of this chapter is to summarize specification of the REST interface for several classes of eScop devices in the Physical layer. It also contains the vision of the service specification in Structured Text Language. As was mentioned before, the application based on the described specification will be tested together with the INCAS mini-pilot simulator, an INCAS mini-pilot and pilot application.

Acknowledgment. This work was partially supported by the University of West Bohemia Grant No. SGS-2010-036.

References

Cheng, B., Cheng, X., & Chen, J. (2015). Lightweight monitoring and control system for coal mine safety using REST style. *ISA transactions*, Vol. 54: 229-239.

Corujo, D., Lebre, M., Gomes, D., & Aguiar, R. (2012). MINDiT: A framework for media independent access to things. *Computer Communications*, Vol. 35/15: 1772-1785.

Fernandez Alcal, M.R., Gonzalez Alonso, I., Garcıa Fuente, M.P.A., & Maestre Torreblanca, J.M. (2010). A Case Study of the Application of UPnP in Robotic and Home Automation Services. *Proceedings of the 7th International Conference on Information Technology: New Generations (INTG'2010)*: 1221–1222.

Fielding, R.T. (2000). REST: Architectural Styles and the Design of Network-based Software Architectures. PhD dissertation, University of California, Irvine.

Kopecky, J., Gomadam, K., & Vitvar, T. (2008). hRESTS: An HTML Microformat for Describing RESTful Web Services. *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT'2008)*: 619–625.

Maciel de Silva, L.A.H., & Hirata, C.M. (2013). Fault-tolerant timestamp-based two-phase commit protocol for RESTful services, *Software: Practice and Experience*, Vol. 43/12.

Newmarch, J. (2005). A RESTful approach: clean UPnP without SOAP. *Proceedings of the 2nd IEEE Conference on Consumer Communications and Networking (CCNC'2005)*: 134–138.

Parra, J., Anwar Hossain, M.A., Uribarren, A., & Jacob, E. (2014). RESTful Discovery and Eventing for Service Provisioning in Assisted Living Environments. *Sensors*, Vol. 14/5: 9227-9246.

Reverb Technologies (2014). Swagger 2.0 Specification. https://github.com/swagger-api/swagger-spec. Accessed 2015.01.27.

Assessment of Communication Protocols for Industrial Devices

Luis Enrique Gonzalez Moctezuma

Tampere University of Technology, Tampere, Finland

luis.gonzalezmoctezuma@tut.fi

Abstract. Industrial automation systems can leverage several benefits of technologies used in traditional Information and Communication Technology (ICT) applications. This work analyzes four communication protocols (DPWS, RESTful Web services, MQTT and CoAP) that currently have relevant momentum, big community and with particular features that if applied to the realm of industrial automation could boost many features like re-configurability, loosely coupled and open systems among others. A comparative table is provided as a result of this comparison. The table can be used by industrial system designers to check the characteristics of each protocol at the moment of deciding which technology to use in the deployment of networked industrial controllers.

1 Introduction

Traditionally industrial automation systems are implemented as a hierarchical structure, which is commonly known as the pyramid of industrial automation. The lowest level is composed by sensors and actuators. They are in charge of retrieving data and acting on the physical process that the industrial system is working on. These sensors and actuators are controlled and monitored by industrial controllers. This is the second level of the pyramid. These controllers can communicate among them or with higher layer components of industrial system. The components that belong to the third level of the pyramid are in charge of scheduling, tracking, reporting and manage the resources that are required for executing the desired production.

The communication between level one (sensors and actuators) and level two (industrial controllers) provides real time requirements. To achieve this goal, fieldbuses are used. These usually implement proprietary protocols which hinders the integration with components from other manufacturers. In industry this is referred as islands of automation. It is important to mention that this document does not intent to cover the communications between level one and level two of the industrial pyramid and therefore, field buses protocols are out of the scope. Instead of that, the aim of this work is to cover the latest communication protocols that can be used to communicate industrial controllers between themselves (horizontal communication) and with higher level components (vertical communication) of the industrial pyramids like Manufacturing Execution System (MES) and Enterprise Resource Planning (ERP) applications.

Industrial controllers are robust pieces of computational hardware which are capable of manipulating inputs (sensors) and outputs (actuators) in a deterministic framework. Their physical construction and components are meant to increase their reliability so that they can operate in the hazardous conditions which are present in industrial facilities like: humidity, temperature, electromagnetic noise and vibration. Usually they are deployed in compact spaces within the industrial facility. In order to achieve a small form factor (SFF) there has to be a sacrifice of computational resources. Therefore, the communication protocols that industrial controllers can handle are those that have been created for resource constrained devices.

The realm of information and communication technologies has many sub domains. Some of them have a very fast dynamics, big communities and great amount of developments going on. For example, web technologies, enterprise applications and Internet of Things (IoT) among others. The result of this, is new communication protocols, open standards, programming libraries, documentation and frameworks that can be reused by others in different domains.

The main goal of this document is to gather, describe, analyze and evaluate communication protocols that have been created in other fields of ICT, which have potential to be used in industrial systems. New features contained in these protocols will open innovative development and research roads in industrial informatics. The market in industrial system integration, demands a more transparent and multi-vendor solutions. In order to achieve this, open communication protocols are needed. Therefore this work will only describe and analyze open communication protocols, which have a potential to be deployed in industrial controllers. The main targets for the moment are: Devices Profile for Web Services (DPWS), RESTful Web Services at device level, Message Queue Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP) and OPC Unified Architecture (OPC UA).

The remaining part of this chapter is organized as follows: section two describes each of these protocols. Section three makes an analysis and assessment of these protocols under the context of their applicability within the field of industrial automation. Finally the conclusions of this work are given in section four.

2 Communication protocols

Representational State Transfer (REST) Web Services.
REST is a software architectural design proposed by Roy Fielding. In REST each source of data is considered as a resource. The most common implementation of REST is by using RESTful Web Services, which rely on the use of HTTP as transport mechanism.

In REST a limited number of methods are defined, which follow a CRUD (Create, Read, Update, Delete) approach. This CRUD fashion maps very well with the HTTP methods: Post, Get, Put, Delete. In practical applications, the most common methods are GET in order to retrieve data from a resource and POST for manipulating a resource or invoking some processing.

It is important to mention that REST is a stateless protocol, i.e. all resources can go over different states, nevertheless the state of each resource has to be handled by the clients and not by the server that interface the resource (Moritz et al., 2010).

The transactions in a RESTful architecture are synchronous request-response transmissions. This implies that a client requesting an action in a server opens socket and holds the socket open until the respond from the server gets completed.

In order to improve the interoperability of a resource exposed as a web service, it is important to describe its functionality and parameters that need to be provided when invoking it. Therefore a service description language is required. REST services can be described by using a simple machine readable service called Web Application Description Language (WADL). WADL is XML-based and describes how to reach the services through HTTP. WADL is widely adopted and therefore exists different implementation of WADL generators.

RESTful implementations can be boosted with semantic descriptions in services and resources by using Semantic Annotations for Web Resources (SA-REST). SA-REST proposes the use of properties to intrusively embed ontological meta-data. In this way a service element can be mapped to an ontological model.

Standard implementations of REST do not provide mechanisms for eventing. In eventing it is expected that the resource/server push messages towards the client. Nevertheless, REST relies on the traditional request-response transactions, where servers are passive elements and they just process data when a client request happens. This is a limitation for the use of REST services in industrial controllers, nevertheless due to the great flexibility of REST services, it is possible to implement eventing mechanisms.

An important aspect of RESTful implementations is that the payload of the messages can be based on the different formats, for example XML or JSON. This is provides great flexibility for the system integrators.

A weak point of REST is its discovery mechanism. REST do not provide any specific framework for web services discovery and just relies on the metadata provided in links for discovering services. So in order to provide discoverability of services, custom mechanisms have to be implemented.

In order to understand the constraints and performance of REST implementations, it is necessary to analyze the behavior of HTTP communications, because the most common implementations of REST architecture are built on top of HTTP.

Devices Profile for Web Services (DPWS).

The DPWS is a communication protocol stack which main objective is to implement Simple Object Access Protocol (SOAP) based Web Services (WS) at device level. This protocol was started by Microsoft and got standardized by OASIS (OASIS, 2009).

When a device implements WSs there is a big amount of features that are inherited for example, mechanisms to secure messages, description of services, WS discovery and publish-subscribe mechanisms. WS have a rich set of standards that cover many areas. DPWS devices are aligned with WS and therefore can leverage these different standards.

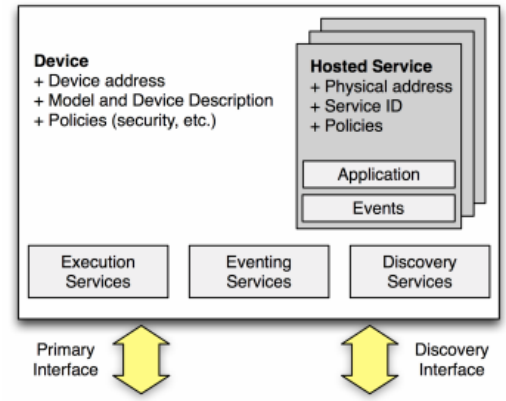


Fig. 1. Device and hosted Services in DPWS specification (Candido Goncalo, et al. 2010)

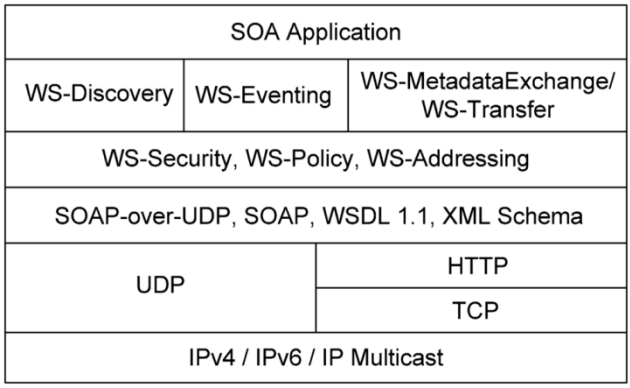


Fig. 2. DPWS stack, from (Zeeb et al., 2007)

Two main elements are defined by the DPWS specification: the device and its hosted services. Discovery and metadata exchange protocols are controlled by the device. The device hosted services encapsulate functions or operations that the device can provide. These elements are shown in the Fig. 1.

As already mentioned DPWS is composed by WS specifications in a very similar structure to the WS architecture. Fig. 2 shows the DPWS stack of protocols and is possible to visualize that underlying protocols give services to upper protocols. By having these specifications in DPWS it is possible to provide the next services (OASIS 2009):

- **Discover**y: by using WS-Discovery, a device can advertise itself and discover other devices in the network. The discovery process works on SOAP over UDP messages. When DPWS servers (devices) are found, the interested party fetches its Web Service Description Language (WSDL file) in order to understand the service that this device provides.

- **Messages addressing**: WS-Addressing provides a mechanism to asynchronously exchange SOAP messages among endpoints.
- **Metadata exchange**: with WS-MetadataExchange it is possible to host services descriptions that can be accessed in form of WSDL and XML schemas.
- **Policies description**: WS-Policy facilitates the description of the service capabilities, requirements and characteristics.
- **Event publish/subscribe**: by using WS-Eventing, devices can subscribe to asynchronous messages which are produced when an event occur.
- **Security**: although WS-Security is not formally part of the profile, it is the most accepted specification to implement security. DPWS security can also be extended by using WS-Trust and WS-SecureConversation (Cheong et al., 2012).

DPWS transfer messages using SOAP, which is XML-based, therefore its verbosity is quite high which translates in big overhead. Indeed this is a disadvantage of using DPWS, its verbosity. Usually the transport layer of SOAP in DPWS is HTTP, therefore it is easy to transfer WS messages on standard local area networks (LAN) and routers do not provide integration barriers. An important feature of DPWS is that it can use UDP or TCP sessions for message delivery, which offers big flexibility depending on the application requirements.

Message Queue Telemetry Transport (MQTT).
MQTT was created by IBM but openly published as a royalty-free license. MQTT is an application protocol that provides a topic-based publish-subscribe mechanism. In practice this means that a device can publish a message to a given topic and any other device or application subscribed to that channel will receive the message. Fig. 3 shows an exemplification of a MQTT-based messaging system.

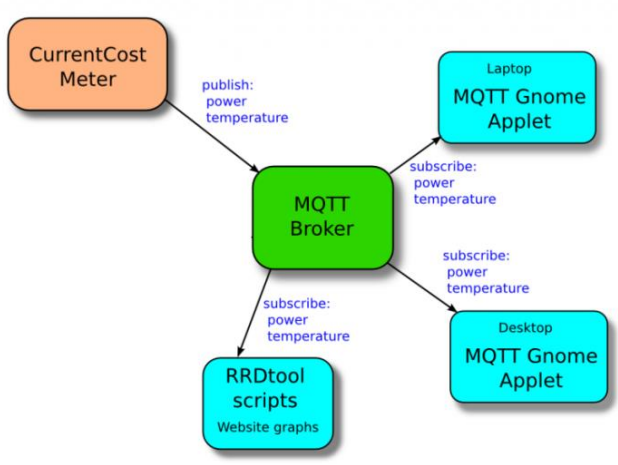


Fig. 3. Example of MQTT messaging implementation (Yeoh, 2009)

This protocol was built having in mind its use in resource-constrained devices, low bandwidth and high latency networks. One advantage of MQTT is its small footprint, therefore it can be deployed in mobile devices or Internet of Things targets. Therefore it can be used with good efficiency in embedded systems like industrial controllers (Thangavel et al., 2014).

An important feature of MQTT is that it works over TCP/IP, like other well known protocols, e.g. Hypertext Transfer Protocol (HTTP). Nevertheless, as mentioned before, it is tuned for resource constrained targets, therefore it has a lower protocol overhead. MQTT is a broker based system instead of a peer to peer system. Even though this approach requires the installation of a broker for its deployment, it provides many other advantages, for example scalability. MQTT-based brokers can support device connections in the order of thousands. Quality of Service (QoS) refers to the fact that messages within a delivery system will have different priorities and treatment. Depending on the class or type of the messages. MQTT allow the possibility to deliver messages with a QoS mechanism:

- Level 0-Fire and forget: In this mode messages are issued and delivery to the end points is not warranted. It reduces the load on the broker but it is unreliable.
- Level 1-At least one time: It ensures that messages are received by the recipient at least one time, but there is a risk that the target receives message more than once.
- Level 2- Exactly once: As its name implies, the broker warranties that message is received by recipient exactly one time. This mode is more heavy in network packages point of view since it needs a four-way handshake mechanism.

MQTT uses a compact binary packet payload. This means that messages do not include properties, headers are compressed and there is much less verbosity than by other protocols, like HTTP. In terms of security MQTT offers possibility to protect privacy of messages by providing support to SSL-encrypted connections. In order to protect brokers and channel topics from not desired parties, authentications with username/password are employed to warranty that valid entities use the system.

Constrained Application Protocol (CoAP).

CoAP is an application protocol. It is a web transfer protocol which is tuned to be deployed in constrained nodes and constrained networks. It is designed for machine-to-machine (M2M) applications like smart energy and building automation.

CoAP is based on a request/response interaction model between endpoints. It has a built-in discovery mechanism which makes it suitable for plug and play applications and adopts important elements from the Web realm like URIs and Internet media types. CoAP is implemented on top of UDP (Shelby et al., 2013).

An important goal of CoAP is not just to provide and efficient and compressed version of HTTP but rather to implement a subset of REST with HTTP having in mind the context of M2M applications. CoAP reduces the implementation complexity and the package sizes of HTTP protocol. Even though CoAP could be used to make more efficient some HTTP interfaces, its main advantages is that it provides built-in mechanism for discovery, multicast and asynchronous message exchanges.

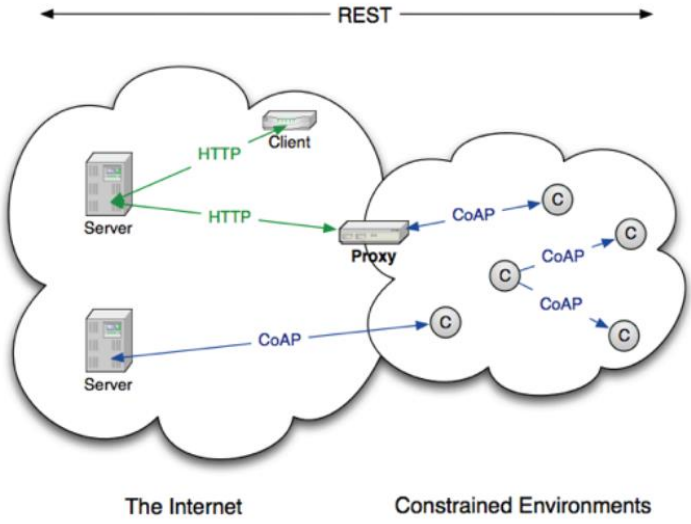


Fig. 4. Interaction of CoAP environment with external ones (Shelby, 2011)

CoAP includes proxy components which allow constrained environments (using CoAP protocol) to interact with standard equipment and infrastructure like PCs or servers which are connected to the internet. All of this using as the communication base a REST architecture (Fig. 4).

It is important to mention that the discovery built-in mechanism of CoAP is not peer to peer, but rather by using a Resource Discovery server. This entity holds information of other nodes and their capabilities. This server is discovered by the clients by using multicast messages.

CoAP can work over different security schemas, on "NoSec" mode the node sends plain messages over UDP/IP. The system can be kept secure by isolating the system from interactions with external agents. The other security modes are achieved by using Datagram Transport Layer Security (DTLS). DTLS is the implementation of Transport Layer Security TLS on top of UDP messages (Shelby et al., 2013).

CoAP makes use of the GET, PUT,POST and DELETE methods in a similar fashion as they are used in HTTP. Nevertheless it can be expanded to new methods by defining new specifications. Here it is important to mention that new methods are not forced to implement request-response in a one to one fashion. For example, the Observe method (which is used for subscribing to events) is a simple request that results in multiple responses.

The way subscriptions are implemented in CoAP is through a method called Observe. By observing a resource, the interested party can retrieve a representation of a state and keep this representation updated. Fig. 5 shows an example of sequence diagram for the observing process for registering and getting updates of a resource (Hartke, 2013).

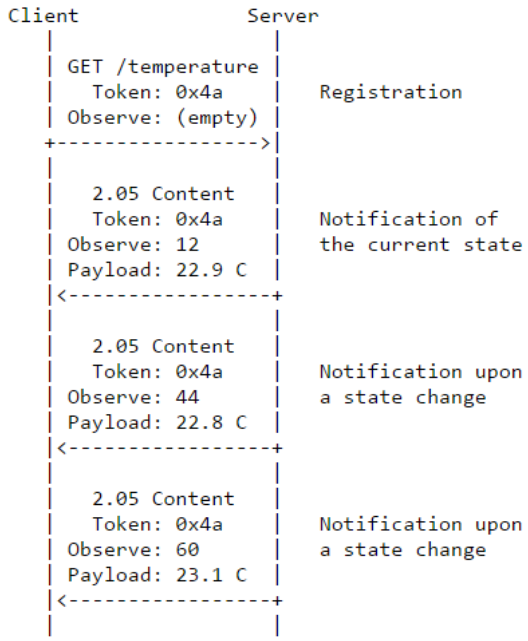


Fig. 5. Sequence diagram for observing a CoAP resource (Shelby 2013, et al.)

The observing process can use intermediaries servers. An intermediary server can work as a middleman, in this way many endpoints can be observing one resource, without actually incrementing the network load on the resource every time a new observer enrolls.

3 Protocols assessment

The main objective of this section is to provide an analysis and discussion comparing the communication protocols previously described. The discussion and comparison is within the scope of using these protocols for communicating industrial controllers with higher layer components of Manufacturing Execution System.

The discussion and comparison will concentrate around seven axes:

- Architecture topology and modelling
- Synchronous/Asynchronous messaging
- Protocol complexity/overhead
- Devices /services discovery
- Devices / services description
- Publish-subscribe mechanisms
- Security

Architecture topology and modelling.
Protocols like DPWS allow implementing Service Oriented Architecture (SOA) designs, where the functionality of a component is encapsulated and exposed to external elements. This is a big advantage for mapping into industrial systems. For example the action of an actuator can be easily exposed as service. More complex set of actions can be encapsulated and exposed as well as individual operations, which can be used by the control units. Therefore DPWS is a good candidate for abstracting industrial systems and interface real equipment and their functionality as SOA components.

On the other hand REST protocols are enablers of Resource-oriented architecture (ROA) designs. The limitations of verbs and methods in RESTful implementations complicate the mapping and modeling of industrial services. This is because the functionality of equipment should be modeled as changes in the state of a resource. REST communications and CoAP share these limitations.

DPWS and REST devices can communicate in a peer to peer fashion. CoAP components can also interact in this one to one manner without intermediaries. Nevertheless it is possible to include CoAP server to let the system communicate with traditional REST components and infrastructure. MQTT is a broker-based topology, which means that needs a server hosting the broker. MQTT does not provide specific methods for executing actions and is more a messaging infrastructure, therefore to map industrial components functionality, the designer has to model within the device the services it offers.

Synchronous/Asynchronous messaging.
As mentioned in the previous section, REST works on top of HTTP transactions, which imply request-response messages. In industrial systems, transactions do not occur immediately, therefore in order to wait for the response of one request, it would be necessary to keep open the connection until the process is completed. This would block the thread of the client as result of a synchronous behavior. By using WS-Adressing, DPWS includes an ID in every message, so that an asynchronous response can be issued with the correspondent request ID. This avoids threads blocking.

MQTT works in asynchronous manner, when a device issues a message it goes to the broker. Then the broker forwards it to the subscribed nodes. In this way the message sender does not wait for a response from the other nodes. CoAP provides great flexibility in this matter. Its interaction model allows devices to execute simple synchronous communications but they can also issue asynchronous messages, that allow the device to execute other tasks without blocking its current thread.

Protocol complexity/overhead.
DPWS is the protocol that has the bigger computational cost due to the fact that it has to process XML-based messages, which is a very verbose format. This results in an elevated overhead, which is not desired for constrained devices or industrial facilities with a high flow rate of messages.

RESTful services overhead should be analyzed from the HTTP protocol perspective which is the technology used to implement REST. It is estimated that 4% of the

message is overhead, this without considering the overhead introduced by the payload format, for example, a XML-based REST service has more overhead than a JSON-based REST service.

CoAP messages are much smaller than REST HTTP packets. Mappings from string to integers are used widely in order to save space. Due to the simplicity of the messages they can be generated and parsed in place which avoids to consume extra RAM in constrained devices. MQTT is also a very simple and efficient messaging system. It is lightweight so it has a lot of momentum in mobile applications. The protocol is so efficient that the overhead can go as little as 2 bytes.

When referring to any of the previously discussed protocols, it is important to notice that if encryption mechanisms are applied on the messages to provide security, then the overhead grows and the complexity of processing the messages in the endpoints grows as well. How much the overhead and processing complexity depends of how light or heavy the security mechanisms are and if they are applied at application or transport layer.

Devices/services discovery.

From the four protocols that are being analyzed DPWS offers the best option in terms of discovery. The built-in discovery mechanism allows devices to have a plug and play behavior. In a MES ecosystem, software applications can recognize new devices, as soon as they are connected to the network. CoAP implementations use resource servers to host the functionality of the devices on the network. When a client needs to discover the resources available in a network it should go to these servers to fetch the available devices. Prior to this, the devices or service have to be registered with the resource server.

MQTT and REST lack of discovery mechanisms. In order to provide discovery in REST deployments, this functionality have to be implemented. Candidates like Universal Plug and Play (UPnP) protocol can be used to complement REST discovery. On the other hand MQTT has more restrictions to implement discovery mechanisms because all the transactions between devices and applications are done through the broker, so it is not possible to implement peer to peer discovery mechanisms, nevertheless discovery routines can be easily achieved by using the broker and announcing the arrival of new devices to the system.

Discovery has a big relevance in industrial systems because it reduces the time spent to reconfigure a system, which in return translates in money savings.

Devices/services descriptions.

Protocols like DPWS and RESTful web services have very complete frameworks to describe the way services must be consumed, their locations and the parameters that they require, by using description languages like WSDL and WADL respectively. They even add the possibility to incorporate semantic data by using notations which link to ontological models.

CoAP makes use of Constrained RESTful Environments (CoRE) Link Format to describe resources, their attributes and their relations to other links. By adding rela-

tions information they go into a semantic description (Shelby 2012). MQTT does not define any description mechanism. Device/services descriptions in combination with automatic discovery are features that can reduce significantly the reconfiguration time of a system. This is a common task in manufacturing systems due to the changes in the scheduled production or the changes in the availability of industrial equipment which can be for example under maintenance when unexpected failures occur.

Publish-subscribe mechanisms.
As it was mentioned in the background section of this document, RESTful implementations do not provide native mechanism for publish-subscribe patterns. This is because the server works in a passive way, just waiting for requests. Nevertheless if the programming environment allows it, it is possible to implement customized publish-subscribe mechanisms by letting devices to have client and server roles at the same time. On the other hand WS-Eventing enables DPWS devices with a very powerful publish-subscribe mechanism where devices and applications can interact in a peer to peer fashion.

MQTT messaging core is based on broker who publishes topic messages to the subscribed entities. In MQTT this mechanism works through an intermediary, the broker. CoAP provides the possibility for observing (subscribers) a resource state directly or through federated servers, giving more flexibility to the deployments. Having intermediary servers between the observers and the resources reduce the network load on the resources improving the scalability of the system.

In industrial systems the use of publish-subscribe mechanisms are highly desired because they allow dynamic engaging or establishment of communication channels between devices and applications. This makes components loosely coupled, because there are no hardcoded end points and therefore it is very easy to migrate from solution vendors and also the configurability of the system is boosted by having dynamic communication channels.

Security.
Cyber security is a very wide branch of Information Technology (IT). In a networked-based system there exist a variety of threats that can be tackled with different protection mechanisms. Depending on the industrial application and it requirements the proper security mechanisms should be selected, among the services they provide, the next ones are the more relevant:

- Authentication
- Authorization
- Data integrity
- Data confidentiality
- Availability
- Intrusion protection

RESTful implementations are based on HTTP, which provides lot of flexibility. The most common way of securing HTTP messages is by using HTTPS which in

practice is the use of HTTP transactions on top of Transport Layer Security / Secure Sockets Layer (TLS/SSL). TLS/SSL are cryptographic protocols to warranty the integrity and confidentiality of messages, by encrypting its payload. The use of certificates also allow the parties to ensure they are talking with the expected endpoint, which represents some degree of authentication. Thanks to the great flexibility of rest messages and its payload it is possible to implement other authentication mechanisms like the combination of username and password.

DPWS can utilize a very rich set of security mechanisms which are defined by the WS-Security suit. WS-Security is very modular and different granularity levels can be defined. For example, a designer can decide to apply encryption to the whole message or just to specific fields of the SOAP message. Messages can be signed to ensure authentication. This security package works mainly at the message level, meaning that the cryptographic algorithms are applied at the payload of the message.

If the flow of messages is highly intensive, the computational requirements can be very costly which does not suit to industrial devices. Just to exemplify, in a benchmark test (Lascelles and Flint, 2006) a target of 2.8 GHz CPU could process 352 messages/second using the WS-Security XML Signature & encryption, compared with 2919 messages/second when using the Transport Layer Security. DPWS can incorporate the WS-Authorization suit for having access control of the services provided by the devices.

MQTT can work on plain mode, without any security mechanism, or the messaging system can encrypt the data packets using SSL, which is a low cost encryption mechanism, by doing this integrity and confidentiality is warranted. MQTT also allows the use of username and password combinations in order to provide authentication mechanisms.

CoAP can work as well under different security levels. When security is desired, the datagrams are protected by using DTLS. DTLS is the encryption of datagrams by using TLS cryptographic methods.

As it is possible to see from the discussion in the previous paragraphs, authorization services, except in DPWS, is not provided. Therefore, in order to have access control, it is necessary to implement authorization mechanisms, which can be simple control access lists. Availability and intrusion protection are also not provided by these protocols. This is because in order to avoid external attackers from accessing the system, it is necessary the use of hardware equipment like firewalls.

Industrial systems are usually and must be carefully isolated from external networks, therefore the use of firewalls or total isolation is desired in order to avoid Denial of Service (DoS) attacks which put in risk the availability of the services that the devices provide.

Considering the case when remote monitoring of the industrial environment is desired, the system has to be properly designed and configured to allow just exit of messages generated at the industrial facilities. The encryption mechanisms must guarantee the data confidentiality and integrity, so that it can arrive safely to the remote monitoring centers.

Table 1. Comparison of different communication protocols for industrial automation

	RESTful	DPWS	MQTT	CoAP
Topology	ROA	SOA	Broker-based	ROA
Federated Components	None	None	Broker needed	CoAP servers can be used
Complexity modeling of industrial devices/services	Medium	Easy	Medium-High	Medium
Messaging	Synchronous	Asynchronous	Asynchronous	Synchronous and Asynchronous
Overhead	Medium	High	Low	Low
Discovery	None but expandable	Native and dynamic	None but broker can announce new devices	Native but static
Device/service description	WADL	WSDL	None	CoRE Link Format
Publish-subscribe	None, but can be implemented	Native	Native	Native (observers)
Publish-subscribe type	None, but can be implemented	Peer to peer	Broker-based	Peer to peer Intermediary server
Security		WS-Security	SSL user/password	DTLS
Security computational load	Low-Medium Depends on the security services	High Depends on the security services	Medium-Low	Medium-Low

Table 1 summarizes and compares all the discussion given in this section. The core points from the seven analyzed axes have been aggregated in this table. The table can be used by industrial system designers to check the characteristics of each protocol when deciding about which technology to use in the deployment of networked industrial controllers.

4 Conclusions

There is a huge number of communication protocols in the ICT world, nevertheless the set of protocols that can be used in industrial communications can be cut to a fraction of them. This work considered to analyze four of them because many of their main features can be used to give extra value in industrial applications. Most specifically, RESTful web services were chosen because of their big popularity and flexibility, DPWS because it is an enabler of SOA architectures where the concept of services can be mapped to functionality offered by industrial equipment. Also because their modularity allows them to offer a very rich set of features which can be chosen depending on the application. MQTT is very lightweight protocol, due to the fact that it relies on a broker. Messages routing and delivery reliability relies on the broker. By doing this, the communication complexity lies on the broker and not in the devices.

Therefore it is feasible to implement it in industrial controllers. It also has a very strong and simple publish-subscribe dynamic. CoAP has a very big momentum in the world of Internet of Things, it is very efficient, simple and has compatibility with standard RESTful implementations.

The theoretical background of these communication protocols was given and they were analyzed under the scope of industrial contexts in seven different axes: Architecture topology and modelling, synchronous/asynchronous messaging, protocol complexity/overhead, devices /services discovery, devices/services description, publish-subscribe mechanisms and security.

Some of these protocols perform better in certain axes, but worse in others. Therefore one-size-fits-all protocol for industrial automation cannot be defined and selection totally depends on requirements of the application. In order to facilitate this, the chapter provides a table resuming where the main differences explicitly exist.

References

Candido Goncalo, F. J., Barata de Oliveiera, J., & Colombo, A.W. (2010). SOA at Device level in the Industrial domain: Assessment of OPC UA and DPWS specifications. *Proceedings of the 8th IEEE International Conference on Industrial Informatics (INDIN-2010)*: 598-603.

Cheong, C.-P., Chatwin, C., & Young, R. (2012). Performance enhancement of WS-security using Participant Domain Name (PDNT). *Proceedings of the International Joint Conference on Computer Science and Software Engineering (JCSSE)*: 213-218.

Elmar, Z., Bobek, A., Bohn, H., & Golatowski, F. (2007). Service-Oriented Architectures for Embedded Systems Using Devices Profile for Web Services. *Proceedingsof the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*: 956-963.

Hartke, K. (2013). Constrained Application Protocol draft-ietf-core-observe-08, *Internet Engineering Task Force (IETF)*.

Lascelles, F., Flint, A. 2006. WS Security performance. http://francoislascelles.syscon.com/node/204424/. Accessed 2015.04.02.

Moritz, G., Zeeb, E., Pruter, S., Golatowski, F., Timmermann, D., & Stoll, R. (2010). Devices Profile for Web Services and the REST. *Proceedings of the 8th IEEE International Conference on Industrial Informatics (INDIN'2010)*: 584-591.

OASIS (2009). Devices Profile for Web Services Version 1.1. *OASIS: Advancing Open Standards for the Global Information Society*. July 2009. http://docs.oasis- open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.html. Accessed 2015.III.26.

Shelby, Z. (2012). Constrained RESTful Environments (CoRE) Link Format. *Internet Engineering Task Force (IETF)*.

Shelby, Z., Hartke, K., & Bormann, C. (2013). Constrained Application Protocol draft-ietf-core-coap-18, *Internet Engineering Task Force (IETF)*.

Shelby, Z. (2011). Embedded Web Services, https://noppa.aalto.fi/noppa/kurssi/t-110.5150/luennot/T-110_5150_embedded_web_services_in_iot.pdf/. Accessed 2015.03.26.

Thangavel, D., Ma, X., Valera, A., Tan, H.-X., & Tan, C.K.-Y. (2014). Performance evaluation of MQTT and CoAP via a common middleware, *Proceedings of the 9th IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP'2014)*: 1-6.

Yeoh, C. (2009). GENOME MQTT Panel Applet 0.1. http://chris.yeoh.info/2009/06/29/gnome-mqtt-applet-01/. Accessed 2015.04.02.

eScop Project Physical Layer Development

An INCAS Mini-pilot Case Study

Pavel Balda and Milan Štětina

University of West Bohemia, Plzeň, Czech Republic

{pbalda, mstetin2}@kky.zcu.cz

Abstract. This chapter describes software for control of production equipment which was developed in the eScop project. The software runs on a network of configurable embedded devices called RTUs (Remote Terminal/Telemetry Units), which are communicating with the upper layers via a RESTful web service interface. The overall development methodology including simulation, which uses model based design technique (software in the loop) is demonstrated on the mini-pilot at INCAS Group premises in Biella, Italy. The mini-pilot includes application of conveyor system.

1 Introduction

This section provides an introduction to the eScop architecture and to basic principles applied along development of the mini-pilot. The consecutive sections of this chapter address details of the development of Physical layer of the mini-pilot, and particularly following topics are addressed:

1. Simulation model of mini-pilot
2. Control system of the simulation model
3. Control system of the mini-pilot
4. Testing of simulation and control system
5. Conclusion and future work

1.1 eScop Architecture

The eScop architecture is based on Service Oriented Architecture (SOA) and is composed of three layers:

- Physical Layer – controls hardware components of the production system.
- Representation Layer – contains the knowledge about the production system in the form of ontology.
- Orchestration Layer – acts as a supervisory control system of the manufacturing process.

The University of West Bohemia (UWB) team is responsible for the Physical Layer (PHL) research and development. Physical layer plays a crucial role in the eScop architecture because it performs hardware and software interface to the technology building blocks (physical devices making up the given production system) and its services are intensively used by the other layers (Representation and Orchestration).

This chapter deals with development of the Physical layer control software. The Architecture of Physical Layer control devices, which are called RTUs (Remote Terminal/Telemetry Units) by the eScop architecture, is described in another chapter within this book (Severa and Pišl), which describes RTUs system software (RESTful web services, Service Manager, etc.).

1.2 Simultaneous Development of All Layers

Organization of the development of all eScop software is pretty complex task. The development should take place in parallel (project partners teams work on different topics) and simultaneously, and it is necessary to start provision of services from the Physical Layer to other layers in such a way. But despite of that there are demanding requests which arose along the project work and the Physical layer requirements deliverable (Camp et al., 2014). One of the most demanding requests is an RTU support of several PLC programming languages described in the IEC 61131-3 standard. In order to Physical Layer software is available in time, it was decided not to develop all physical layer software simultaneously but to start with the development of a web service toolkit, see (Severa and Pišl, 2014).

In this phase of development, the REX control system for real-time control of technology building blocks is used (REX Controls, 2014), which implements a PLC software supporting Function Block Diagram (FBD) (it is compatible with the Matlab/Simulink), Sequential Function Chart (SFC) and C-like programming language (simplified C language, see the REXLANG block in (REX Controls, 2014)). The REX implementations of FBD and SFC are with a few exceptions compatible with IEC 61131-3 and the C-like language substitutes Structured Text (ST) language also from IEC 61131-3.

In the next development phase, the REX control system can be replaced in simple cases by a runtime based on ST which is being developed. This approach makes possible to accelerate the development of Physical Layer software as well as to provide data to upper layers.

1.3 INCAS Mini-pilot Application

The whole methodology described in the previous subsection is demonstrated on the case study of first SOA control system of the mini-pilot application in INCAS group project partner, Biella, Italy. The schema of the mini-pilot conveyor system is depicted in Fig. 1 which was published in (Cincera, 2014). Note that the INCAS mini-pilot application is a part of the whole INCAS pilot application which should be put into operation later. The details of the pilot application are described in (Cincera et al., 2014).

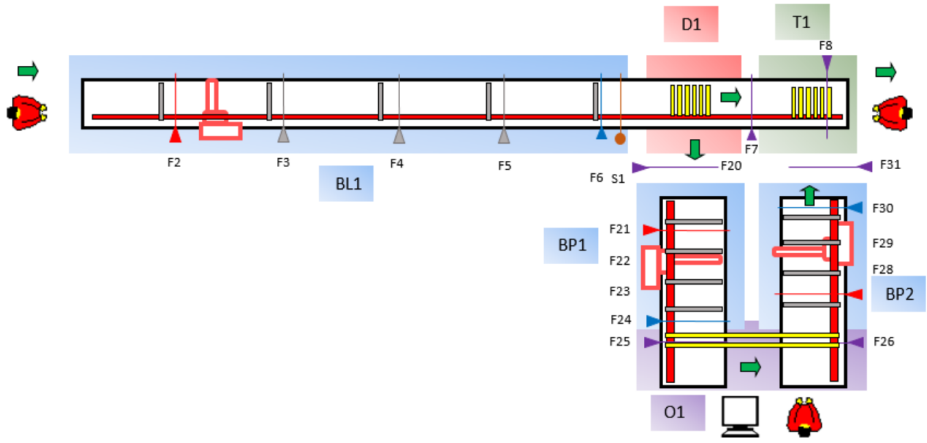


Fig. 1. Layout of the INCAS mini-pilot application conveyor system

The Figure 1 contains the following technology building blocks (modules) (Cincera, 2014):

BL1 **Buffer and incoming point**. It allows incoming of cases and their accumulation before being sent to the picking station O1.
D1 **Decision making module**. It transfers cases from BL1 to BP1 or to T1 expelling them from the system.
BP1 **Picking buffer**. It accumulates cases for the picking station O1.
O1 **Picking station**. It contains the selected case for picking activity.
BP2 **Picking buffer**. It accumulates cases for transfer to the output module T1.
T1 **Output module**. It transfers cases, which come from BP2 or from D1, out of the system.

Cases can be moved on the conveyor system in the direction of thick green arrows in Fig. 1. Buffers BL1, BP1 and BP2 are modular, consisting of smaller *Photocell – Clutch roll* elements.

A photocell is depicted as a thin line ended by a small triangle and it is marked by the letter F and the photocell number. Each photocell indicates its status (presence of a case in front of it) by single digital input. Signals from gray colored photocells are wired internally and are not available in the control system.

A clutch roll is represented by a thin gray filled rectangle which precedes a corresponding photocell (in the direction of movement). Each clutch roll can be connected to or disconnected from the driving belt by single digital output. Outputs to clutch rolls corresponding to gray photocells are also connected internally (not available for the control system).

When the given photocell signals the presence of the case and the following (in the movement direction) photocell – clutch roll elements are full, the clutch roll should be disconnected from the drive belt. This functionality should be implemented by the

control system with the exception of the photocell – clutch roll elements which are not available in the control system where this behavior is hard-wired.

The decision making module D1, output module T1 and picking station O1 contains *Caterpillars* which are represented by yellow filled parallel rectangles in Fig. 1. Caterpillars are controlled by a pair of digital outputs. First output moves up and down the caterpillar construction, the second one starts and stops movement of the caterpillar belt in the direction perpendicular to the main direction of conveyors.

First experience with the mini-pilot application was acquired in May 2014 along the technical workshop in Biella, which was attended by the majority of project partners. In advance of the workshop, the simulation system of mini-pilot conveyors as well as the first control system have been developed. Both these systems have been adapted into compliance with reality during the workshop.

2 Simulation Model

There are many software tools for simulation of discrete processes which are suitable for simulation in logistics including conveyors, e.g. AnyLogic (Anylogic, 2015), Arena (Rockwell Automation, 2015), Witness (Lanner, 2015), etc. Powerful simulation tools are built-in into CAD packages, e.g. Plant Simulation (Siemens, 2015) belongs to Tecnomatix portfolio which is built upon Siemens PLM, or Plant Works (Javelin, 2015) which is fully integrated add-in to SolidWorks by Dassault Systemes, etc. A very powerful tool is also Modelica Simulation Environments (Modelica, 2015) and its libraries. However, such powerful tools are not necessary for eScop architecture software development. More important feature is the possibility to develop a real-time simulator which sufficiently corresponds to the physical conveyor behavior from the control point of view. Therefore a real-time simulator of mini-pilot has been developed in REX using the Model Based Design technique. Everyone who is familiar with Matlab-Simulink (The Mathworks, 2014) is able to understand REX function block diagrams because of REX compatibility with Simulink.

2.1 Overview of the Simulation Model

The whole mini-pilot simulation is performed by a simulation task in Fig. 2. The task consists of six subsystems (big green blocks), each of them simulating a single technology module described in subsection 1.3.

Inputs and outputs of these subsystems have the following meanings:

SIM_RES	Reset the simulation to the initial state
TR_IN	Transfer in the incoming case
TR_OUT	Transfer out the containing case
TR_LFT	Transfer in from the left (T1 only)
CP_UP	Caterpillar is up (in the upper position)
CP_LFT	Caterpillar is moving to the left
CP_RHT	Caterpillar is moving to the right
F_PRE	Photocell preceding the module (O1 only)

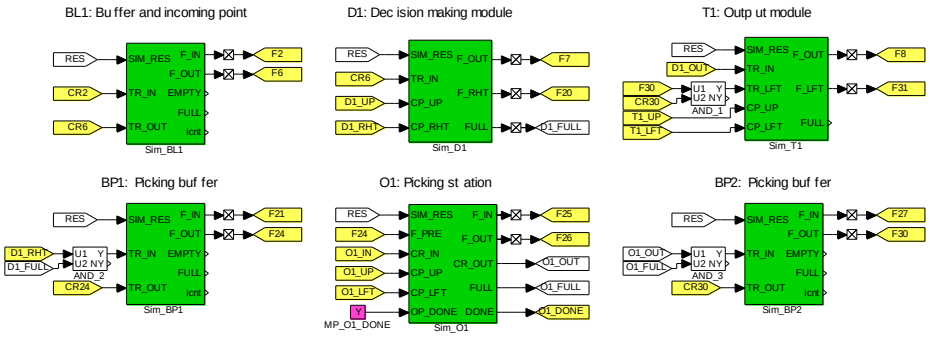


Fig. 2. Mini-pilot simulation model overview

OP_DONE	(Picking) operation is done (O1 only)
F_IN	Input photocell of the given module
F_OUT	Output photocell of the given module
DONE	Operation is done (O1 only)
EMTPY	Flag indicating that there is no case on the buffer
FULL	Flag indicating that the module/buffer is full
icnt	integer number of cases on the buffer

All logical (digital, binary) signals use the positive logic. The logical 1 (or true, on) means that a case is in front of the photocell, a clutch roll is connected to the drive belt (the conveyor is moving), a caterpillar is up or moving, etc. Similarly, the logical 0 (or false, off) means that there is no case in front of the photocell, a clutch roll is disconnected from the drive belt (the conveyor is stopped), a caterpillar is down or stopped, etc.

Simulation subsystems are interconnected by From and Goto blocks. These blocks with the same Tag connect the corresponding output with one or more inputs without the necessity to draw wires. Used tags correspond to photocells (e.g. F6, F20, F24), clutch rolls (e.g. CR6 or CR24, adjacent to photocells F6 or F24), commands from the control system (e.g. D1_UP, O1_LFT, etc., see section 4) and RES for simulation reset. Yellow filled signal of From and Goto blocks are communicated between the simulation model and real-time control task.

Blocks marked as ⊠ are instances of the LPBRK (Loop Break) block which is important for determination of block execution order of function block diagrams containing loops and for avoiding hazards. See the LPBRK block documentation in (REX Controls, 2014) and problems with algebraic loops in (The Mathworks, 2014) for more details.

The remaining part of this section describes the simulation of buffers BL1, BP1 and BP2. Simulations of the other modules (D1, O2 and T1) are done similarly and are omitted for brevity.

2.2 Simulation of Buffers

Buffers BL1, BP1 and BP2 differ by physical size, but from simulation and control point of view they are very similar. The only difference is maximum number of containing cases. BL1 buffer can hold up to five cases, while buffers BP1 and BP2 only four. That is why only simulation of BL1 is presented here. Fig. 3 exhibits content of simulation subsystem Sim_BL1 (from Fig. 2). Simulation of five photocell – clutch roll elements is covered by subsystems FC_CR_1, … , FC_CR_5. They are described in the next subsection. The TR_IN command is lead to the first photocell – clutch roll subsystem, the TR_OUT is lead to the last photocell – clutch roll subsystem. Similarly input photocell of the buffer is read from first photocell – clutch roll subsystem, output photocell is read from last photocell – clutch roll subsystem. The buffer FULL resp. EMPTY flags are constructed from signals FLT_F (filtered FULL) of each photocell – clutch roll subsystems using eight-input blocks for logical AND (ANDOCT) resp. logical OR (OROCT). The count of cases icnt is obtained from eight-input addition block (ADDOCT) returning a real (double precision) value converted to integer by RTOI block. Note that simulation subsystems of buffers BP1 and BP2 differ from BL1 by containing only four photocell–clutch roll elements instead of five.

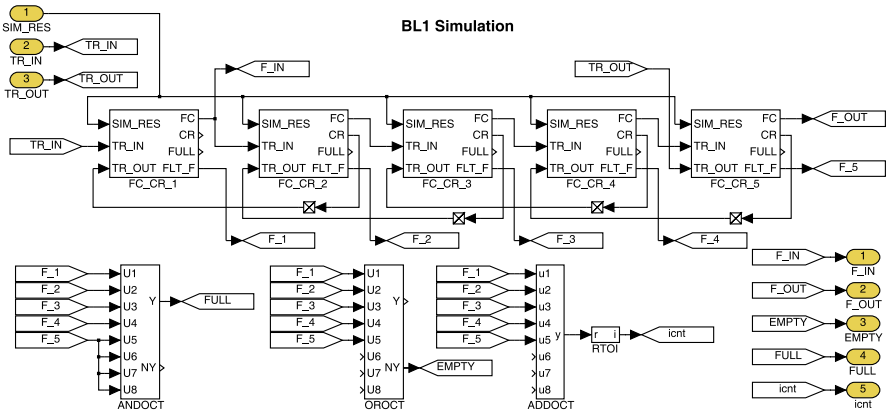


Fig. 3. Simulation of Buffer BL1

2.3 Simulation of a Single Photocell – Clutch Roll Element

The simulation subsystem of the photocell – clutch roll element is depicted in Fig. 4. The most important block in this diagram is the ATMT block implementing a finite-state automaton (see (REX Controls, 2014) for details) with up to 16 states. Its algorithm can be configured in Sequential Function Chart language of IEC 61131-3. The current SFC step is indicated by logical 1 on one of the outputs Q0,…,Q15. The SFC transition conditions are prepared externally and are connected to the inputs C0,…,C15 of the ATMT block. The particular configuration of this block is shown in Fig. 5. Simulation starts from the initial state Q0: Empty. Signal C0: TR_IN causes a transition to state Q1: Filling.

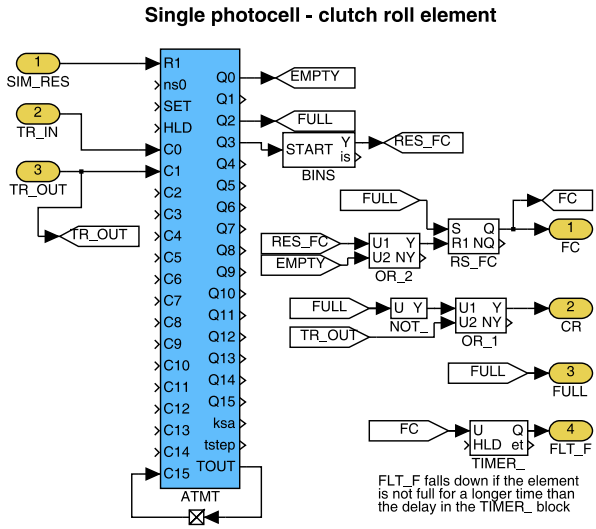


Fig. 4. Simulation of single Photocell – Clutch Roll element

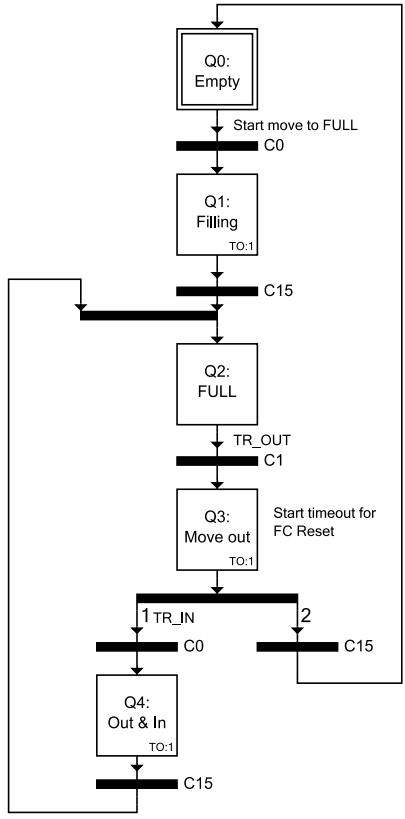


Fig. 5. SFC Algorithm of ATMT block (from Fig. 4)

After one second, the filling timeout (TO: 1, see bottom right corner of this state in Fig. 5), the TOUT output of the ATMT block is set to logical 1. This value is lead to the C15 input in the next execution of the ATMT block, which causes the transition to the step Q2: FULL. This step remains active until setting the C1: TR_OUT signal. Then follows the transition to the step Q3: Move out where one second timeout is awaited. If the signal CO: TR_IN arrives sooner, transition to the step Q4: Out&In is performed and the photocell – clutch roll element is immediately filled again. After the 1 second timeout in the step Q4, the Q2: FULL step is reached. Otherwise after elapsing the timeout at step Q3 the step Q0: Empty is achieved.

The other blocks in Fig. 4 controls the logic of output signals. The photocell FC activity is controlled by a RS flip-flop. It is set when the full state is reached and it is reset after the 0.9 second delay generated by the BINS block elapses or after reaching the empty state. The clutch roll CR is moving when the state is not full or transfer out signal arrives. A special role described in Fig. 4 is played by the FLT_F (filtered full) signal. It remains true when signals TR_IN and TR_OUT are active (almost) simultaneously because the timer delay is a little bit longer then inactivity of the simulated photocell.

Now, let us explain the behavior of the sequence of photocell – clutch roll subsystems in Fig. 3. The photocell signal FC (i.e. the module contains a case) of the previous subsystem triggers the transfer in operation of the next subsystem. The movement will be started if the next photocell – clutch roll module is not full. When the buffer is empty, the TR_IN command applied to the first photocell – clutch roll module causes the movement of entered case until the last photocell – clutch roll module. Repeating the TR_IN command (without any TR_OUT command) fills the whole buffer. Applying TR_OUT command to the last photocell – clutch roll subsystem which is in the FULL state will transfer out the case and sets temporarily the CR signal. This signal is fed back to the previous photocell – clutch roll subsystem which is also transferred out and this mechanism is repeated until the first empty photocell – clutch roll is reached upstream.

3 Control System of the Simulator

The mini-pilot simulator control system has a similar structure as the simulator described in section 2. The control task (Fig. 6) also consists of six subsystems (big blue blocks). Each of them controls a single simulated module.

Some inputs and outputs have the same meaning as those in simulation, the others are:

OM_EN	Output module of this module is enabled
RM_EN	Module to the right of this module is enabled
F_RHT	Photocell to the right of the given module
F_LFT	Photocell from the left of the given module
IN_FULL	Preceding module the given module is full
LFT_FULL	Preceding module for movement to the left is full
LFT_IN	Transfer in the incoming case by movement to the left

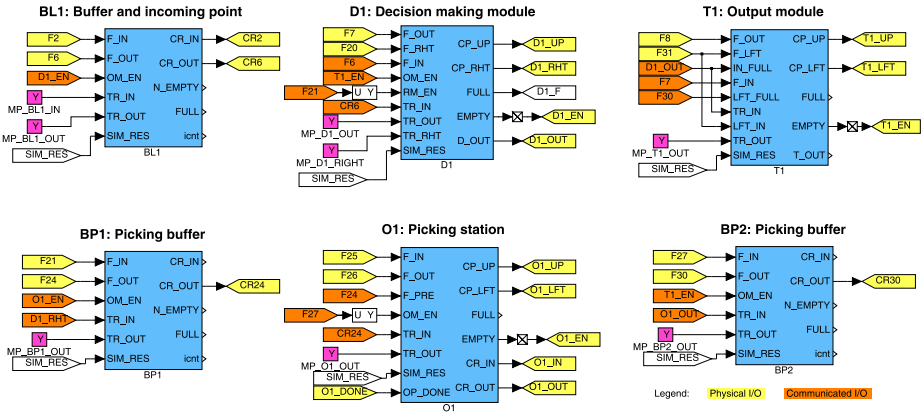


Fig. 6. Mini-pilot Simulator Control System

CR_IN	Activation command to input clutch roll
CR_OUT	Activation command to output clutch roll
TR_RHT	Transfer out to the right (D1 only)
N_EMPTY	Flag indicating that the buffer is not empty
D_OUT	Case is being transferred out (D1 only)
T_OUT	Case is being transferred out (T1 only)

The rest of this section describes the control algorithm of buffers BL1, BP1 and BP2. Control algorithms of the other modules (D1, O2 and T1) are done similarly and they are omitted for brevity.

3.1 Control Algorithms of Buffers

Similarly to subsection 2.2, all buffers BL1, BP1 and BP2 are controlled by the same control algorithm. Without loss of generality it is sufficient to describe the control algorithm of BL1 which is depicted in Fig. 7.

Again, the main block is the ATMT block, whose SFC algorithm is shown in Fig. 8. The initial step Q0 is a transient state, much time is spent in the step Q1: Done where transfer out and transfer in commands are awaited. If both commands come in the same time instant, the transfer out command has the priority. Both commands are decomposed into three steps, start, wait and end.

If the transfer out command is enabled (output photocell must be active and output module must be enabled and TR_OUT is active) the ATMT block enters the step Q2: out start. Here the case is moving out (deactivation of the output photocell) is expected. Then two transient steps are sequentially gone through and the Q1: Done step is reached through the Q0: Init. The step Q3: out wait is intended for future reserve and is not used in present implementation of the control algorithm. Similarly, if the transfer in command is enabled (input photocell is not active, i.e. the buffer is not full, and TR_IN is active) the ATMT block enters the step Q5: in start. Here the case is moving in and activation of the input photocell is expected. Then again two transient steps are sequentially gone through and the Q1: Done step is reached through Q0: Init.

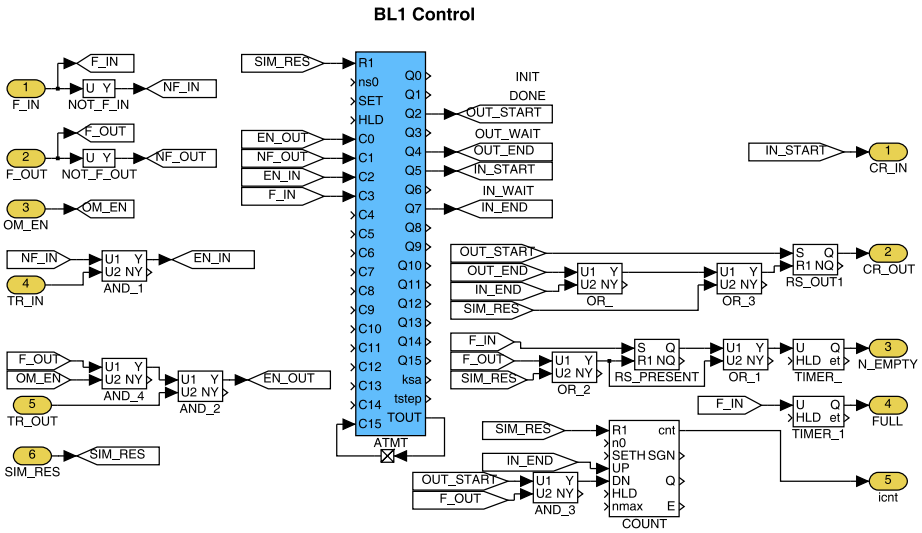


Fig. 7. Control algorithm of buffer BL1

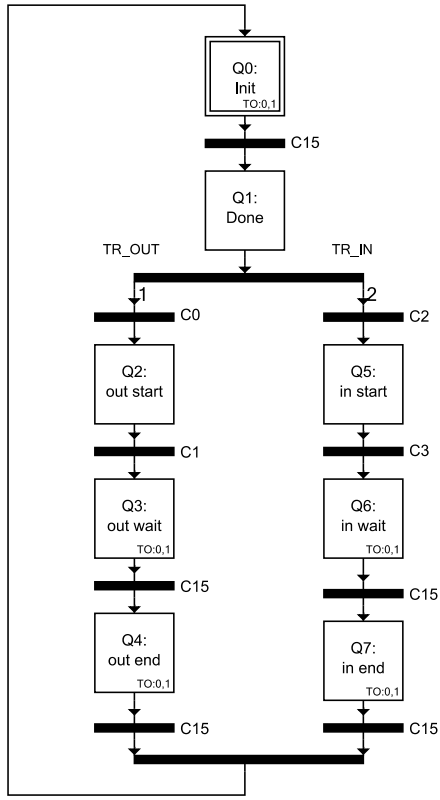


Fig. 8. SFC Algorithm of ATMT block (from Fig. 7)

Remaining blocks in Fig. 7 generate activation of clutch rolls CR_IN and CR_OUT. Outputs N_EMPTY, FULL and icnt (number of cases on the buffer conveyor) are generated here only for visualization purposes.

4 Control System of the Mini-Pilot

As already mentioned above, the Mini-Pilot control system was tested at the INCAS site in Biella, Italy, in May 2014. The real application was temporarily equipped with six RTUs, each RTU controlled one technology module. RTUs were based on a single board computer ALIX 2D13 (PC Engines, 2014) and remote inputs/outputs (I/O) from the X20 family (Bernecker & Rainer, 2012). I/Os were connected to ALIX computer through the bus controller X20BC0083 communicating via Ethernet POWERLINK industrial protocol. Control algorithm of the whole mini-pilot consists of a set of control algorithms of all interconnected RTUs. Each RTU's control algorithm was created by following procedure:

1. Extract the subsystem (including adjacent From and Goto blocks) corresponding to particular technology module from the mini-pilot simulator control system (Fig. 6).
2. Add connections to remote inputs and outputs.
3. Add communication with neighboring RTUs.

The result of this procedure for the buffer BL1 is depicted in Fig. 9. Inputs from the physical buffer are read by the EPL driver via the EPL__DI2 signal which returns all bits from the corresponding X20 family input module as an integer value. This value is decomposed to bit signals by the bitwise demultiplexer BDOCT_DI2 (instance of the BDOCT function block). Signals from photocells F2 and F6 were connected to fourth and fifth bits (zero based index) and had negative logic (that is why they are preceded by NOT blocks).

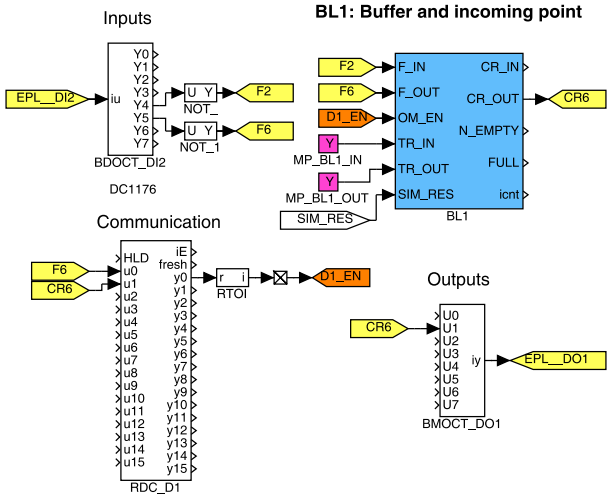


Fig. 9. Control algorithm of the RTU for BL1

Outputs to the physical buffer are handled similarly via EPL__DO1 integer signal, so bitwise multiplexer BMOCT function block should be used. Unlike simulation the CR2 clutch roll is not connected because the case in transferred in the buffer automatically when it is put in front of the photocell F2.

Communication among RTUs will be provided by Representation Layer (RPL) and Orchestration Layer (ORL) in the final eScop solution. Because development of these layers takes place simultaneously with the development of physical layer, it is necessary to substitute RPL and ORL by internal communication. Blocks RDC (Remote Data Communication) communicating through UDP/IP protocol has been used for this purpose. RDC blocks communicate peer to peer, inputs of one block are sent to outputs of its counterpart and vice versa. It follows from the Fig. 6 that in the case of BL1 RTU only the communication with D1 RTU is necessary. Values of F6 and CR6 should be sent to D1 RTU and only the signal D1_EN should be received from D1 RTU. Note that yellow filled signals are read/written locally in each RTU, the orange filled signals are signals which have to be communicated from/to another RTU.

5 Testing

First tests of mini-pilot simulator as well as its control system were done during the initial development before the technical workshop in Biella. Several design errors was identified during the workshop when comparing the behavior of the simulated mini-pilot with the real physical application. Despite this, the first eScop control system has been successfully put into operation. Moreover, the first version of RESTful web services based human machine interface (HMI, visualization and operator commands) by the project partner ICONICS was also successfully tested, see more details in (Severa and Pišl, 2014).

After returning back, much effort has been devoted to improvements of simulation as well as control. The following testing plan has been prepared and divided into 6 steps:

1. **Mini-pilot simulation: 1 RTU, PHL, HMI**
   This step is already implemented. All configuration runs in a single RTU, control subsystems communicate via internal variables.
2. **Pilot simulation: 1 RTU, PHL, HMI**
   This step will be prepared until eScop Warsaw meeting in middle October 2014. The main difference from the previous step is the extension to the entire pilot application.
3. **Pilot simulation: 1 RTU, PHL, HMI, RPL, ORL**
   Based on the previous step, to which first versions of RPL and ORL for INCAS pilot are added. Control subsystems communicate locally via RESTful Web services in a service manager.
4. **Pilot simulation: n RTUs, PHL, HMI, RPL, ORL**
   Based on the previous step, the application is split to all necessary RTUs. Control algorithms communicate via RESTful Web services through Ethernet/Internet.

5. **Pilot control: n RTUs, PHL, HMI**
   This step will be implemented in real INCAS pilot application. RTUs will communicate via RDC blocks (using UDP/IP) just for simplifying the control system debugging.
6. **Pilot control: n RTUs, PHL, HMI, RPL, ORL**
   This is final state of the INCAS pilot application. All layers are implemented and correspond to eScop architecture.

Each of the proposed steps should be properly tested and the results reported.

6 Conclusions and Future Work

The development of the eScop project solution resembles the ancient causality dilemma: which came first, the chicken or the egg? A relatively complex multi-layer software should be developed using the same software itself.

This chapter proposes a step-by-step solution from the point of view of the lowest layer – the Physical Layer, which controls the factory floor technology building blocks and performs data using RESTful web services to upper layers. The second issue is a large geographical distance between project partners, which requires to find effective ways how to develop individual software component as much as possible in partner's localities. Therefore, the Model Based Design principle has been used and real-time simulation has been developed.

In order to maintain the clarity of text, the overall approach has been demonstrated on the particular application – the mini-pilot application in INCAS Group, Biella, Italy. This approach can be easily adapted to a different type of application.

The current development covers the first two steps from section 5. The following modifications of the Physical Layer implementation are planned for the next four steps:

- Change of target hardware of RTUs to Raspberry Pi (Raspberry Pi, 2015) and UniPi I/O modules (UniPi, 2015).
- Full implementation of Service Manager based on RESTful web services, see (Severa and Pišl, 2014) for more details.
- Implementation of IEC 61131-3 compliant Structured Text language called from Service Manager.

References

AnyLogic (2015). Multimethod Simulation Software. http://www.anylogic.com. Accessed 2015.II.25.

Bernecker & Rainer (2015). X20 System http://www.br-automation.com/cs/products/io-systems/x20-system. Accessed 2015.II.25.

Camp R., Cincera, G., Pala, S., Balda, P.(Pavel), Balda, P.(Petr), Zyskowski, D., Olivieri, M., Strzelczak, S., Gonzalez, L., & Pišl, R. (2014). Physical layer requirements, *Deliverable D3.2*, EU Grant no.332946, eScop: Embedded systems Service-based Control for Open manufacturing and Process automation.

Cincera, G. (2014). Mini Pilot – INCAS. *eScop project internal presentation*.
Cincera, G., Soppera, F., & Camp, R. (2014). Scenario and Requirement for Pilots, Technical Document 1. *The eScop Project internal document*.
Javelin Technologies (2015). PlantWorks A Complete 3D Plant Design & Visualization Solution. http://www.javelin-tech.com/main/products/plantworks.htm. Accessed 2015.II.25.
Lanner Witness (2015). http://www.lanner.com/en/witness.cfm. Accessed 2015.II.25.
Modelica (2015). Modelica Simulation Environments. https://www.modelica.org. Accessed 2015.II.25.
PC Engines (2014). ALIX2D13 System Board. http://www.pcengines.ch/alix2d13.htm. Accessed 2015.II.25.
Raspberry Pi (2015). http://www.raspberrypi.org. Accessed 2015.II.25.
REX Controls (2014). Function blocks of the REX Control System – reference manual. http://www.rexcontrols.com. Accessed 2015.II.25.
Arena (2015). Arena Simulation Software, Rockwell Automation. https://www.arenasimulation.com. Accessed 2015.II.25.
Severa, O., & Pišl, R. (2014). REST enabled physical devices in service oriented architecture, *Materials of the eScop Consortium Meeting*, Warsaw, 2014.X.23-24, Poland.
Siemens - Plant Simulation Software(2015). http://www.plm.automation.siemens.com/en_us/products/tecnomatix/free-trial/plant-simulation.shtml. Accessed 2015.II.25.
The Mathworks (2014). Simulink help. http://www.mathworks.com/help/simulink/index.html. Accessed 2015.II.25.
UniPi (2015). The Universal Raspberry Pi add-on board. http://unipi.technology. Accessed 2015.II.25.

Cyber Security for Web Service-Based Industrial Devices

Luis Enrique Gonzalez Moctezuma

Tampere University of Technology, Tampere, Finland

luis.gonzalezmoctezuma@tut.fi

Abstract. Industrial systems encompass critical infrastructure like production systems or public services. It is a high priority to secure these assets from cyberattacks because they can compromise the integrity of persons. For long time, security by obscurity was used to protect industrial automation systems, nevertheless nowadays this is not possible anymore since these systems use open communication protocols like Web Services. This work identifies different security threads in industrial systems, analyzes the possibility to implement security mechanisms in devices that implement Web Services and also provides a set of decision diagrams to select the architectural components and security protocols depending on the system requirements, characteristics and context.

1 Introduction

The adoption of Web Services (WS) in industrial systems promises a new level of reconfigurability, scalability and interoperability among industrial devices and applications. However, it also opens doors to many of the threats and cyberattacks known in the IT world. The aim of this work is to identify the mechanisms, techniques and knowledge, existing in traditional IT cyber security, which can be applied to improve the cyber security of an industrial system. This work considers strictly the fact that the Web Services are deployed at device level, thus the selection of security mechanisms and architectures has to fit with the resource constrained nature of this application.

It is important to understand the system that needs to be protected; therefore this study extracts the main characteristics and requirements of industrial systems deployed under centralized and distributed layouts. Security mechanisms used in IT cannot be applied seamlessly at the device level. Thus a feasibility study is done around implementing security in devices that use Web Services for communication, and specifically in devices that run the Devices Profile for Web Services (DPWS). Different security mechanisms are assessed and chosen, so that they fit to the constraints imposed by devices themselves and by the industrial environment where they are deployed.

Without any doubt one of the most outstanding properties of the DPWS stack is its alignment and interoperability with Web Service specifications. The current work exploits this benefit by describing how different protocols from the Web Services Security suite can be used to provide a variety of security services within the target system.

This work explains how many threats known in the IT world are applicable to an industrial system that uses Web Services for monitoring. However, an important contribution of the study is to point the vulnerabilities that arise when Web Services are used at device level in an industrial environment. It is not possible to provide a security framework that covers all possible threats for all types of industrial systems. The security framework has to be tailored and configured depending on the application needs. The final contribution of this work is a set of guidelines, in the form of decision diagrams, which help designers to select the architectural components and security protocols depending on the requirements and characteristics of a given system.

The remaining part of the chapter looks as follows. Section 2 provides background for understanding DPWS and Web Services Security. Section 3 analyses the constraints of implementing security mechanisms in DPWS and identifies different security threats in industrial applications. Section 4 provides a set of decision diagrams used to select architectural components and security protocols, depending on the application requirements and context. Finally conclusions of this work are provided.

2 Background

DPWS.

DPWS is a device-targeted WS protocol stack profile started by Microsoft. Since June 2009 it became a standard of the OASIS. DPWS enables WS capabilities on resource-constrained devices. Such capabilities include, secure invocation of WS operations, description and dynamic discovery of WS and mechanisms to subscribe and receive events from WS. It should be highlighted that devices that implement the DPWS are fully aligned with the WS technology. At the time of this writing DPWS 1.1 is the standard supported by OASIS (OASIS, 2009).

The DPWS specification defines two main elements: the device and its hosted services. The devices are important for the discovery and metadata exchange protocols. The services, which are hosted in the device, encapsulate the functionalities that the device can provide (Candido Goncalo et al., 2010).

DPWS is composed by WS specifications in a similar structure to the WS architecture, previously reviewed. Fig. 1 shows the stack of protocols that composes the DPWS. Due to them it is possible to provide next services (OASIS, 2009):

- ***Discovery***: by using WS-Discovery, a device can advertise itself and discover other devices in the network.
- ***Messages addressing***: WS-Addressing provides a mechanism to asynchronously exchange Simple Object Access Protocol (SOAP) messages among endpoints. SOAP is an XML based format for exchanging messages between Web Services.
- ***Metadata exchange***: with WS-MetadataExchange it is possible to host services descriptions that can be accessed in form of WSDL and XML schemas.
- ***Policies description***: WS-Policy facilitates the description of the service capabilities, requirements and characteristics.
- ***Event publish/subscribe***: by using WS-Eventing, devices can subscribe to asynchronous messages which are produced when an event occurs.

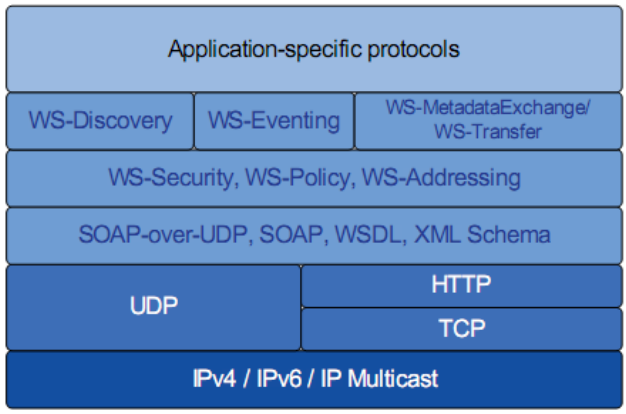


Fig. 1. Protocol stack of DPWS (Zeeb, et al. 2007)

- ***Security***: although WS-Security is not formally part of the profile, it is the most accepted specification to implement security. DPWS security can also be extended by using WS-Trust and WS-SecureConversation.

Web Services Security.

WS security relies pretty much in technologies that secure XML messages since they are the core format of SOAP messages. WS standards, which employ these technologies, have been developed by different organizations to cover gaps between current security standards and WS (Douligeris, C. and Serpanos, D., 2007). These standards are grouped within the Security category of the WS standards stack. The technologies and WS standards that allow implementation of security mechanisms in WS are shown in Figure 2.

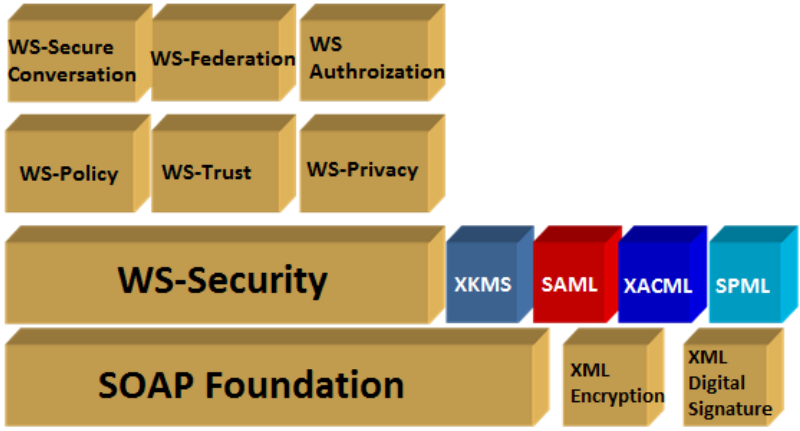


Fig. 2. WS Security Standards

Web Services Security Standards.
It is a set of SOAP extensions used to secure WS under different contexts and security models. The standards allow to implement security services without affecting functional characteristics of a system. The main purpose is to allow secure interoperability of systems, preserving essential characteristics of WS, like platform independence, composability, flexibility, modularity. The key characteristic of WS Security standards is end-to-end security of communication, i.e. a message can pass through different participants preserving specific security requirements for all of them.

The WS Security suite is formed by individual and interrelated specifications which are associated in a layered architecture (Fig. 2), thus allowing to mix and implement just the needed components. The WS Security suite main standards are:

- ***WS-Security***: is the basis for other security specifications as it provides functions that enable integrity and confidentiality to WS messages. It relies heavily on the use of XML Signature and XML encryption and it uses other technologies like Security Assertion Markup Language (SAML) assertions. SAML is an XML-based format for exchanging data concerning authentication and authorization between parties. Some branches of this standard are provided in (InnoQ, 2007):
  - SOAP message Security: provides flexible support for multiple security token formats, trust domains, signature formats and encryption methods.
  - SAML Token Profile**:** describes how SAML assertions should be used with the WS-Security: SOAP message Security specification.
  - X.509 Certificate Token Profile**:** defines how to use the X.509 technology with the WS-Security: SOAP message Security specification. It is common in scenarios where business partners need to exchange information.
- ***WS-Policy*****:** defines a model and syntax to express and communicate general purpose policies related to a WS.
- ***WS-Trust***: describes a model that allows establishment of trust by using security tokens. It provides extensions on top of WS-Security that allows the entities to exchange, issue, renew and validate security tokens. Typically security tokens are expressed as SAML tokens. WS-Trust is a key enabler of federated security system.
- ***WS-Privacy***: specifies a model to integrate statements on privacy in WS; makes use of WS-Policy to communicate them; describes how WS-Trust is used to evaluate privacy claims and uses WS-Security to support cryptographic methods/tokens.
- ***WS-SecureConversation***: used to establish a secure session. This helps to reduce overhead of exchanging multiple independent messages, as a unique Security Context Token (SCT) is used to authenticate messages exchanged in the session. This protocol improves performance of systems that require constant messaging.
- ***WS-Federation*****:** describes federated model for WS in different trust domains by merged use of WS-Security, WS-SecureConversation and WS-Trust. The standard specifies protocols for managing transactions and defines security precautions in a federated system. It is composable with WS standards of Reliable Messaging.
- ***WS-Authorization***: indicates how claims contained in tokens should be structured, interpreted and processed in order to permit or deny access to WS. It is used as an access control mechanism.

In the common model, when implementing the WS Security standards suite, a WS provider requires a set of claims to be fulfilled before providing the service. This set of claims is expressed as a policy by using WS-Policy. Common claims would be identity credentials or access permissions. Then the service consumer requests those claims to a security authority, the STS, which is exposed as a WS. The STS returns the requested claims in the form of a security token. This security token is signed by the STS and can be verified by the WS provider.

DPWS Security.
The DPWS specification has a section dedicated to description of how security should be implemented. The main guidelines and points are summarized in this subsection. It is also given a review of the research efforts that attempt to implement security mechanisms in the DPWS stack.

DPWS 1.1 Standard
In the security section of the DPWS 1.1 standard, it is recommended a minimal set of technologies and mechanisms to enable secure communications between devices and clients. The standard is flexible and allows the devices to support alternative security mechanisms and profiles in order to address the application requirements. The use of other security profiles may by indicated by means of WSDL or policies (using WS-Policy). It is responsibility of the implementer to take care of composability of standards for each security profile in order to achieve interoperability(OASIS, 2009).

The DPWS security focuses on services like: authentication of devices, integrity and confidentiality of messages exchanged between the devices. To achieve this, the standard proposes model composed by two parts:

- ***Message-level signatures for UDP WS-Discovery traffic***: with WS-Discovery Compact Signatures a client can proof integrity of the received discovery messages and proof the identity signed by the sender. This mechanism requires the use of WS-Security to generate the cryptographic signature.
- ***Transport-level encryption for metadata and control traffic***: in order to establish a secure point-to-point communication channel TLS/SSL is used. This allows authenticating the identity of the participants and ensures the integrity and confidentiality of the exchanged messages. Once the authentication has been proved, the HTTPS protocol is used to exchange the messages.

The standard recommends that devices use x.509.v3 certificates as their credential. This credential can be used to sign WS-Discovery messages and for establishing the TLS/SSL channel. Devices can authenticate a client by using their x.509.v3 certificates obtained in the TLS/SSL negotiation or can require their username/password credentials once the secure channel has been established. The general diagram of this model is illustrated in Figure 3.

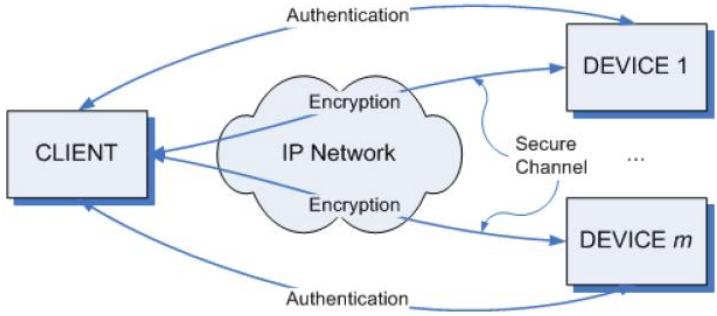


Fig. 3. DPWS Standard security model (OASIS, 2009)

Information Security in Industrial Systems.

Industrial systems are evolving constantly because of the adoption of new communication technologies. These changes raise new vulnerabilities and requirements that the field of industrial information security has to cope with. Industrial deployments used to be protected under the schema of *Security by Obscurity* where the industrial facility used to be isolated from untrusted networks and technical knowledge about the protocols and equipment was unknown by the attackers. These assumptions are not anymore true for industrial systems because nowadays they use open communication standards and use public infrastructures for data transmission like Internet (Dzung, et al. 2005). *Defense-in-depth* is used nowadays to provide security in industrial facilities by using different layers and mechanisms. The assets exposed in outer zones are less secure that those in inner zones. In this way, critical infrastructure like industrial components should be located within the inner zones (Polk et al., 2010). Fig. 4 exemplifies how different are previously described security mechanisms that can be used for protection of an industrial system.

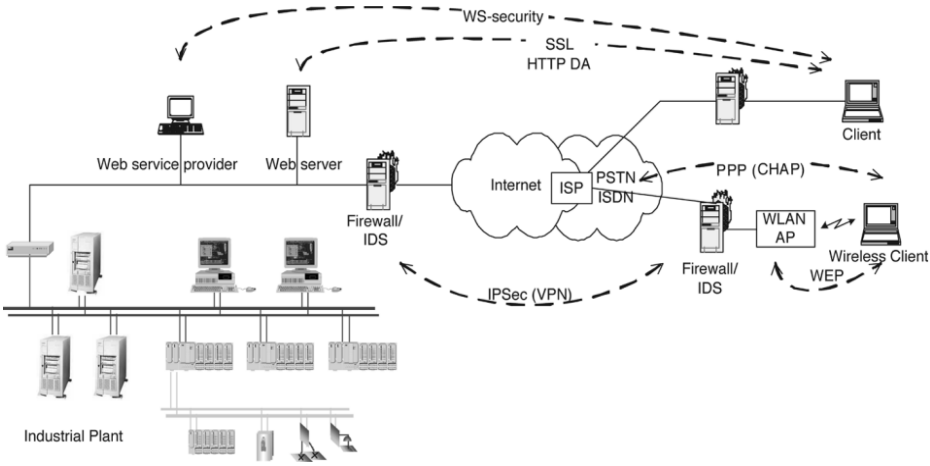


Fig. 4. Typical security mechanisms in an industrial system (Dzung et al., 2005)

3 Security assessment

Industrial system scope.
In order to perform a security assessment, it is necessary to define the scope of the industrial system that is intended to be protected. For this purpose the protected system is targeted, which is composed of industrial devices, which are all together interfaced and controlled by an assigned industrial controller that implements the DPWS stack.

The devices can be monitored either locally or remotely. In case if the monitoring process is running remotely, a gateway or a layer 3 router is in charge of routing the messages through Internet to the monitoring server. The monitoring server concentrates, consolidates, stores and processes data in order to offer some meaningful information about monitored system, through a web-based interface (or other), which can be accessed by the users through a web browser. This framework is illustrated in Fig. 5.

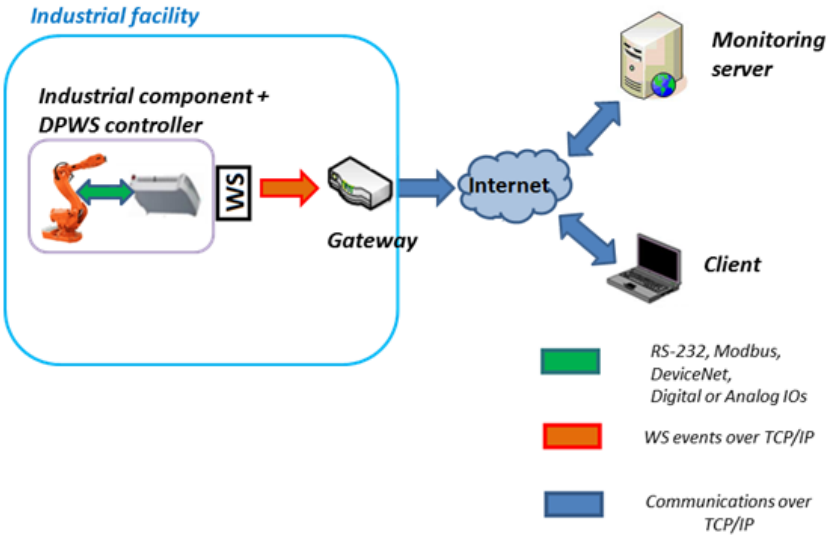


Fig. 5. Monitoring of industrial ambients with Web Services

Security Requirements.
The security requirements listed in this section enable the secure transmission of monitoring data from an industrial facility to a remote server, using Internet as the communication network. The security requirements have been chosen considering the purpose of allowing system scalability and flexibility and having in mind the performance impact of using encryption algorithms in embedded devices, like e.g. the DPWS controllers.

Relevance of security services.
When monitoring an industrial system it is a primordial requirement that the data is protected against any alteration during its transmission. If the monitoring data is tampered with false values the system can suffer serious damages. For example, if a monitoring center is checking the pressure of a gas tank and the value of the measurement is modified by an attacker, there is the possibility to have dangerous pressure levels but the monitoring center might perceive that the tank pressure is at safety values. That is why data integrity is necessary in industrial communication systems for control and monitoring purposes.

In this application the monitoring events are generated at device level. This implies that the data sensitivity of the messages is rather low. Usually the monitoring events will have values concerning the readings of a sensor or the state of some piece of equipment. Very likely this is out of the interest for an attacker, unless the reading represents some critical piece of information that can be used for further malicious purposes.

If data or event aggregation is done within the industrial environment then the data sensitivity increases, as it was discussed in the previous section. In case protection against the disclosure of the monitoring events data during the transmission is required, confidentiality services must be implemented.

From the monitoring server perspective, it is important to authenticate the sources of the events, in order to process and analyze the data properly. The authentication can be done through simple credentials like username/password or more sophisticated ones like SAML tokens could be used.

If the industrial devices allow remote subscriptions to their events, it is necessary that the device authenticate the service requester. Username/password or other tokens can be used for this.

The monitoring events do not contain commands or instructions, they contain passive data in the sense that they do not request any kind of control over resources. This way the authorization service is not required at the monitoring service. Similarly to authentication, if the industrial devices allow remote subscription to their events, it might be required that they can authorize which granularity or type of events the service requester can subscribe. In this case the DPWS devices must implement authorization mechanisms.

Non-repudiation service is useful to demonstrate that transactions did occur. In a monitoring service that involves third parties this can be useful to resolve disputes among the parties. Having this property can be useful when doing audits of the system transactions. Non-repudiation can be used as well as a mean to provide proof of integrity and origin of data.

Table 1 summarizes the relevance of the security services for the purpose of monitoring industrial ambients. The table includes their priority, the conditions in which they should be applied, the implementation side (monitoring server / DPWS device) and an estimate of their computational demand[1].

[1] The computational demand estimation was done considering the data from the reviewed literature

Table 1. Security services relevance

Security service	Priority	Apply in case of	Implemented at	Computation demand
Integrity	1	• Always	DPWS device	Low
Authentication	1	• Server side needs to authenticate source • DPWS controllers offer monitoring services to external network entities	Monitoring server/DPWS device	Low/ Medium
Confidentiality	2	• Event/data aggregation is done in the industrial environment.	DPWS device	Heavy
Authorization	3	• DPWS controllers offer monitoring services to external network entities	DPWS device	Low/ Medium
Non repudiation	4	• Interaction with third parties • Auditability required	Monitoring server/ DPWS device	Heavy

DPWS Devices Constraints and Capabilities.

Cryptographic algorithms can be very resource demanding even for computers with standard computational power, therefore a wrong selection of the cryptographic suite can decrease considerably the performance of the embedded device. The processors speed and the amount of RAM memory vary depending of the device, but considering as reference the current DPWS-based controller they can be around 8MB of memory for data execution and storage and running at 55MHz.

With these resources, a DPWS-based controller must be able to work as a control unit and as web server simultaneously. As a control unit it should storage and run the control/monitoring logic of the process/device it interfaces. As a web server, the controller must decrypt and parse the messages it receives. The output messages should be encrypted, wrapped in the SOAP format and sent to the proper recipient.

The average size of a monitoring event is around 800 bytes. This is due to the fact that XML is a verbosity format. This size of message is comparable to the size of messages (1KB) used in the test realized by (Delamer and Martinez Lastra 2006), described in chapter II, where different cryptographic functions where tested in a networked industrial controller with similar computational capabilities (100MHz, 2MB RAM) to this application controller. Thus, their results can be applicable to this case.

In their test, depending on the selected cryptographic function, the encryption/decryption total time of a 1KB message can range from 132 ms (using AES-128 bit) to 737 ms (using Rijindael 2.0 – 128 bit). The creation and encryption of a hash of the message can range from 20 ms (using MD5 – 128 bit) to 9 401 ms (using PSS Signature Generate 1024 bit).

The results obtained in (Gupta, et al. 2005) proof how embedded devices with even smaller computational capabilities (around 10 MHz, 50 KB RAM), can implement a secure web server using SSL with public key encryption algorithms. With this is possible to provide proof of origin, integrity and authenticity to the messages.

Threats Identification.
The application of monitoring an industrial environment with WSs is exposed to different types of threats. The impact that those threats can have on the industrial infrastructure can vary but they can be as severe as putting in risk the integrity of workers and public services. The objective of this section is to identify the different kind of threats that can affect the information assets of the monitoring system or the system itself.

Spoofing Attack.
The spoofing attacks are based on creating a fake identity in order to gain access to the network and to get some privileges over targeted resources. Such action affects directly the authentication and authorization services. However, if the attack succeeds and the attacker participates in the communications, other threats may arise. For example the integrity and confidentiality of the messages that goes through the attacker can be easily compromised or the attacker can perform some Denial of Service (DoS) attack against some specific target. DoS attacks attempts to overflow a system by sending several requests in such a way that the target is unavailable to the legitimate requests.

Monitoring a system with DPWS devices possess an application level vulnerability that can be exploited to execute these attacks. Recapitulating, when a new DPWS device joins a network it goes into a discovery process. In this process a probe match multicast is sent to all the devices within the network in order to discover the available devices, their hosted services and how to access them, all this information contained within the WSDL file. The easiness to perform this discovery and the lack of confidentiality protection uncovered by the DPWS standard (DPWS standard covers integrity and authentication of discovery messages) leaves a door open for the attacker to join the network and get to know all the available devices and their hosted services, after all WSs are meant to be easy accessible.

Eavesdropping Attack.
The principle of an eavesdropping attack is to intercept the messages exchanged between two endpoints. Once the message has been intercepted the information within it can be disclosed or even modified, which would represent a potential hazardous exposure for the system stability. The most popular schema for eavesdropping is the so called “Man in the middle”, i.e. when an attacker makes the endpoints believing that they are communicating directly between them, while actually all the communication messages are flowing through the attacker.

If the industrial network uses a hub to connect the devices each to other, the data can be easily disclosed by sniffing in the network. This kind of attack is highly probable in a system that uses DPWS, because of the discovery vulnerability, which was already discussed in the previous subsection. Eavesdropping threat also arises if the monitoring system includes multiple intermediaries; in that case if sensitive fields of the SOAP message are not properly encrypted, then the data can be disclosed by non-intended intermediaries.

Logon Abuse Attack.
This threat is possible if the credentials to access a resource are obtained by an attacker, or the authentication and authorization mechanisms bypassed. Once the attacker bypasses them, it has unauthorized access to the resources. This can imply that other threats would be exploited once the attacker manages to access the network or resource.

It is more likely that the organization insiders manage to get to know valuable information that would allow them to do logon abuse attacks. For example operators can get to know the device configuration passwords and make malicious or accidental changes in the device settings.

DoS Attack.
It is important to remember that in order for a DPWS device to work as a service provider, it must listen for service requests. This is done by running an embedded web server which receives the WSs requests. Those requests are transported over HTTP. Thus, a DPWS device is exposed to the same DoS attacks as other HTTP servers would be.

The working principle of DoS attacks is very simple, the objective is to make a resource unavailable to its legitimate users by saturating the target with communication requests. Once the target collapses it is unable to provide its service. Networked assets are very vulnerable to this kind of attacks and in order to prevent them, a traffic blocking component is required, like firewalls, switches, routers, proxy appliances or server clusters. In cyber security, DoS have shown to be a very simple to execute and effective attack.

Application Level Attack.
Application level attacks are possible due to flaws or vulnerabilities in the application layer of the OSI model. For example, in the case DPWS devices or other component of the monitoring architecture has to validate XML signatures for authentication, there would be threat to suffer Web Service Signature Overflow. In this attack a big amount of nested signatures are placed within the SOAP message. The unit attempting to validate the nested signatures would have to decrypt each by each until it reaches an overflowing point. Again, the low memory of DPWS devices makes them susceptible to this type of attacks.

Radio Jamming.
Radio jamming attacks occur by injecting radio signals in a wireless medium. The objective of these signals is to obstruct or block the wireless communication either by inference or by decreasing the signal to noise ratio of the communication channel. This attack falls in the category of DoS attacks but is mentioned separately to remark this kind of attacks on wireless channels.

Table 2 summarizes the threats identified in this section, the security services they affect, the monitoring layout (centralized, distributed) which are susceptible and the agents (insiders, outsiders) that are likely to execute those threats.

Table 2. Threats in the monitoring of industrial environments with WSs.

Threat	Affected security service	System layout	Agent
Spoofing attack	•Authentication •Authorization •Indirectly others[2]	•Centralized •Distributed[1]	•Insiders •Outsiders[1]
Eavesdropping attack	•Confidentiality •Integrity •Indirectly others[2]	•Centralized	•Insiders
Logon abuse attack	•Authentication •Authorization	•Centralized •Distributed	•Insiders
DoS attack	•Service availability	•Centralized •Distributed[1]	•Insiders •Outsiders[1]
Application level attack	•Service availability •Indirectly others [2]	•Centralized •Distributed[1]	•Insiders •Outsiders[1]
Radio jamming	•Service availability	•Distributed	•Outsiders

[1] In case remote subscriptions to events are allowed
[2] Other security services are not directly affected by the threat but can indirectly be exploited

4 Secure Framework for WS-based Industrial Devices

There is no unique solution that provides security for all the threats of all systems. Therefore the used suite of architectural components and security protocols has to be selected depending on the target system. This section proposes two decision diagrams for choosing architectural components and security protocols depending on the system requirements, its components limitations, the available security mechanisms and the threats to which the system is susceptible.

4.1 Architectural components

The architectural components of the framework must be arranged in a layered topology in order to provide Defense-in-depth security. This is was found as a requirement for protecting industrial infrastructure in previous analyses. As it was already mentioned, the main objective of the architectural components and their arrangement is to protect the assets and grant their availability.

DPWS devices are intended to use a gateway for communicating the monitoring data through internet. It is important that a firewall is used in that gateway and its configuration is properly set.

Fig. 6 exhibits a decision diagram in order to facilitate the selection of the framework architectural components. The blocks in blue color decide the flow of the diagram while the blocks in yellow color set a component or recommendation to include into the framework.

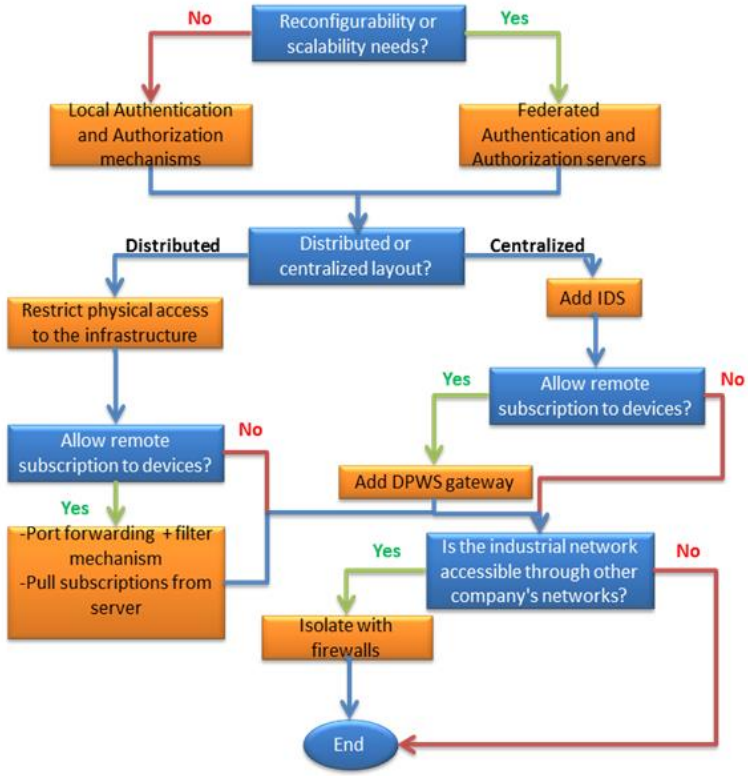


Fig. 6. Decision diagram for choosing the architectural components of the secure framework

4.2 Security protocols

Security protocols are used to provide data integrity and confidentiality in the exchange of monitoring events. They also offer authentication and authorization services in order to allow or deny access to DPWS devices and their hosted monitoring services. From the security mechanisms assessment section, it is possible to notice that there is a wide variety of security protocols. They should be selected depending on the application requirements and threats to which the system is susceptible.

Once the security protocols have been selected, it is important for the implementer to ensure interoperability among them. This can be a serious technical problem, if the security protocols are not flexible or not meant to be interoperable with other protocols at different OSI layers. A big advantage of using components from the WS Security suite, is that they have been designed with composability in mind, thus interoperability is easily achieved between them.

Fig. 7 shows a decision diagram that allows choosing the security protocols depending on the industrial application characteristics. The blocks in blue color decide the flow of the diagram while the blocks in yellow color define the security protocol that must be implemented and included by the framework.

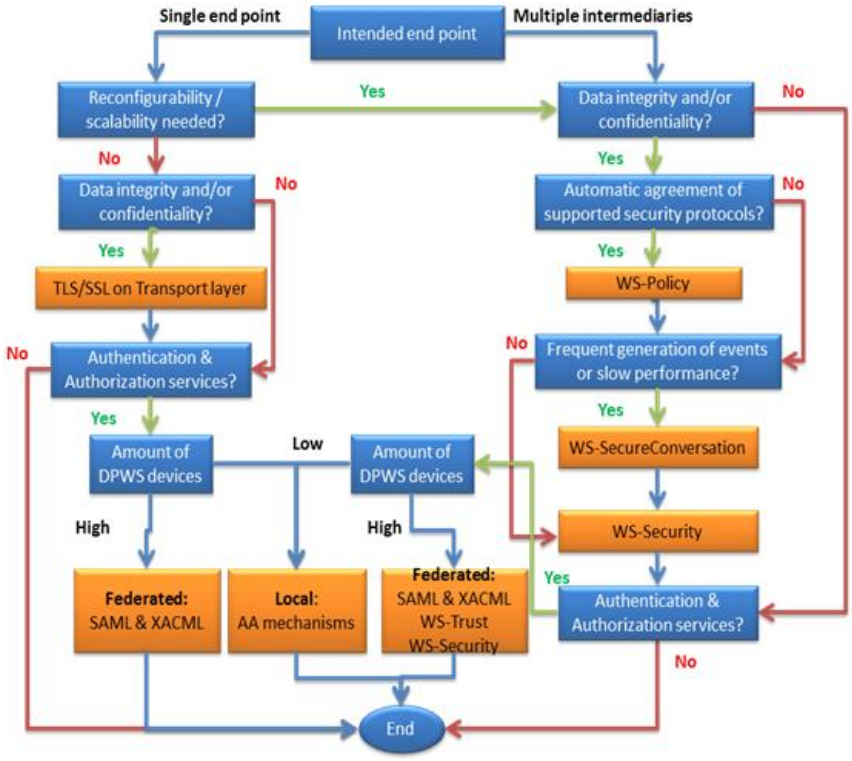


Fig. 7. Decision diagram for choosing the security protocols of the secure framework

5 Conclusions

Industrial systems encompass critical infrastructure like production systems or public services. It is a first priority to secure the all the assets of the industrial systems (physical and logical), since the compromise of any of them can risk the integrity of the personnel and the system itself.

Security requirements of industrial systems differ from those of traditional IT applications. For instance, for the specific case of industrial monitoring, it was found that data integrity and authentication services have the highest priority. Therefore, the selection of security mechanisms should be done with the aim of warranting these two services.

Data confidentiality and authorization are second priority services; but of course, it all depends on the application. If the data coming out from the industrial environment has high sensitivity, then data confidentiality is necessary. If monitoring service providers will subscribe to the industrial devices events, then authorization services are required.

There is a huge amount of security algorithms, frameworks and mechanisms that are used in IT cyber security. Unfortunately security-related processes are computa-

tionally demanding and therefore it is not possible to use them seamlessly at device level due to the fact that DPWS devices/controllers are resource constrained. This fact forces designers to select and tune the security mechanisms according to the application needs. Fortunately, the DPWS stack is enough flexible in security terms, since it allows the use of different technologies and it is highly interoperable with security mechanisms offered by the WS Security suite.

Among all the security mechanisms candidates that can be used in DPWS devices, those who empower a federated security schema are desired. Federated security allows to isolate the authentication and authorization processes from the DPWS devices, by using a trusted server to perform these tasks. This is wished since these tasks can be computationally demanding for resource constrained devices.

In cyber security it is not possible to have a framework (security components and protocols) that fits and covers all the systems against an undetermined number of threats. It is necessary to arrange and tune the security framework depending on the characteristics and needs of the industrial system that needs to be protected. This document proposes a set of guidelines, in the form of decision diagrams, that will help the designer to select the architectural components and security protocols for a given industrial application.

References

Candido Goncalo, F. J., Barata de Oliveiera, J., & Colombo, A.W. (2010). SOA at Device level in the Industrial domain: Assessment of OPC UA and DPWS specifications. *Proceedings of the 8th IEEE International Conference on Industrial Informatics (INDIN-2010)*: 598-603.

Delamer, I. M., & Martinez Lastra, J.L. (2006). Information Security for Reconfigurable Manufacturing Systems using Networked Embedded Controllers. *Proceedings of the International Conference lnformation Control Problems in Manufacturing (INCOM'2006)*: 129-134.

Douligeris, C., Serpanos, D. (2007). Network Security - Current Status and Future Directions. John Wiley & Sons, 2007.

Dzung, D., Naedele, M., von Hoff, T.P., & Crevatin, M.. (2005). Security for Industrial Communication Systems. *Proceedings of the IEEE*, Vol. 93/6: 1152-1177.

Elmar, Z., Bobek, A., Bohn, H., & Golatowski, F. (2007). Service-Oriented Architectures for Embedded Systems Using Devices Profile for Web Services. *Proceedingsof the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*: 956-963.

Gupta, V., Wurm, M., Zhu Y., Millard, M., Fung, S., Gura, N., Eberle, H., Shantz, S.C. (2005). Sizzle: A standards-base end-to-end Security Architecture for Embedded Internet. *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications(PerCom 2005)*: 247-257.

InnoQ (2007). Web Services Standards Overview. *InnoQ Resources*. http://www.innoq.com/resources/ws-standards-poster. Accessed 2015.III.23.

OASIS (2009). Devices Profile for Web Services Version 1.1. *OASIS: Advancing Open Standards for the Global Information Society*. July 2009. http://docs.oasis- open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.html. Accessed 2015.III.26.

Wade, P., Malkiewicz, P., & Novak, J. (2010). Industrial Cyber Security: From the Perspective of the Power Sector. *DEF CON 2010*.

About Authors

Pavel Balda
He is senior researcher at NTIS (New Technologies for the Information Society) European Center of Excellence at the Faculty of Applied Sciences, University of West Bohemia, Plzeň, Czech Republic. Before joining the university, he worked at R&D departments of Škoda and ZAT, and he was a co-founder of Easy Control and REX Controls. His research interests include real-time control and simulation, industrial communications and embedded systems. He teaches Control Software Tools, Information and Control Systems and Embedded Control Systems.

Tomasz Borowiecki
He obtained MSc (1998) in Robotics and Software Engineering, and PhD (2007) in Automatics and Robotics – Automation of Discrete Processes, both degrees from the Technical University of Poznań. His research and teaching activities include: New programming languages and technologies, Production flow management and optimization, Constraints Logic Programming Languages, development of ERP systems in the area of logistics, Methods of implementation for Business Intelligence systems.

Roberto S. Camp
He obtained his BSc in Electronic Systems Engineering from the Tecnológico de Monterrey (ITESM) in 2005. After graduation he worked as Control Engineer in Nexon Automation. He continued his studies at the Tampere University of Technology, from where he obtained MSc in Machine Automation. In TUT, he worked as a Research Assistant at the Department of Production Engineering, being involved in the EU RTD projects. Between both, field and academic experience, he is now an R&D Engineer at FluidHouse Oy company.

Luca Fumagalli
He is post-doc researcher at the Department of Management, Economics and Industrial Engineering of Politecnico di Milano, since 2007. He is adjunct professor for the course Management of Industrial Plant at Politecnico di Milano. He has collaborated in national and EU projects, concerning ICT for manufacturing, maintenance management and industrial engineering. He is presently in charge of research at Observatory Technology and Services for Maintenance (TeSeM) and vice-director of the Master Megmi (Executive Master in Industrial Maintenance Management) program.

Marco Garetti

He is full professor of Industrial Technology at the Politecnico di Milano, and director of the Executive Master on Management of Industrial Maintenance. He started work in the R&D department of Alfa Romeo, then joined Politecnico di Milano, where he teaches Maintenance Management and Design and Management of Production Systems. He is author or co-author of 8 books and over 200 papers. His research focuses on maintenance management, sustainable manufacturing, manufacturing automation, production management and product-lifecycle management. Married, with two sons, he likes motor-biking, sailing, skiing and traveling.

Luis Enrique Gonzalez Moctezuma

He received BSc degree in Mechatronics, with specialization in Control from the Instituto Tecnologico y de Estudio Superiores de Monterrey, campus Ciudad de Mexico, in 2007. In 2011 he obtained MSc degree in Factory Automation from the Tampere University of Technology in Finland. He is currently working there as a project researcher in the Factory Automation Systems and Technologies Lab. His research interests are in the fields of industrial informatics and artificial intelligence. He loves tacos and football.

Sergii Iarovyi

He received the MSc degree (with Distinction) in Electromechanics from the National Technical University - Kharkiv Polytechnic Institute, Kharkiv, Ukraine, in 2011, and the MSc degree in Factory Automation from the Tampere University of Technology (2014). Since 2012 he works at the Factory Automation Systems and Technologies Lab of Tampere University of Technology as Project Researcher. His research interests are in application of semantic web services, cyber-physical systems and enterprise integration for factory automation.

Pavel Lederbuch

He is researcher and software development engineer in ICONICS Europe. He is with his company for 11 years, and works for the European development team, located mainly in Plzeň, Czech Republic. He has MSc degree in Computer Science from the University of West Bohemia, Plzeň, Czech Republic. His work is focused on design and architecture of user interfaces and configuration tools for HMI/SCADA visualizations. Currently he is leading the effort in EU research projects in ICONICS Europe.

Andrei Lobov

He is lecturing at the Tampere University of Technology. He received his PhD in Formal Methods of Factory Automation, in 2008. He holds BSc in Computer and System Engineering from the Tallinn University of Technology (2001). Then, he continued his education at the Tampere University of Technology and received MSc in Automation Engineering (2004). His research interests include development of architectures, methodologies and technologies for manufacturing systems. He is a technical coordinator of the eScop project.

Stanisław Marciniak

He is full professor of business management at Warsaw University of Technology. Currently he is the Head of Department for Production Systems Organization. His research interests include assessment of production systems efficiency, managerial accounting, sustainability, strategic management and management control. He authored and co-authored 10 books and over 140 papers. He teaches courses on economics, financial and managerial accounting, and runs PhD seminars. He used to work as advisor or non-executive director to international corporations, including blue chips (e.g. Daewoo, PKN Orlen).

Elisa Negri

She is PhD student at the Department of Management, Economics and Industrial Engineering of the Politecnico di Milano. Her research fields are ontology-based engineering and industrial systems modeling. Her PhD thesis concerns the study and development of ontology-based advanced manufacturing solutions, in particular for open manufacturing. She graduated in Management Engineering at Politecnico di Milano, with the thesis developed in the FIR Institute for Operations Management, Aachen, Germany. She received a double BSc degree in Mechanical and Production Engineering from the Politecnico di Milano and the Tongji University in Shanghai (PRC), with a focus on production processes.

Roman Pišl

He is a researcher at NTIS (New Technologies for the Information Society) European Center of Excellence at the Faculty of Applied Sciences, University of West Bohemia, Plzeň, Czech Republic. His research interests include real-time control and simulation, embedded systems and real-time communication. His recent work is focused on developments for real-time operating systems and Linux based control systems.

Borja Ramis Ferrer
He received BSc degree in Electrical Engineering from the Universidad de las Islas Baleares, Islas Baleares, Spain, in 2011, and the MSc degree (with distinction) in Factory Automation from the Tampere University of Technology, Tampere, Finland, in 2013. He is currently working towards his PhD degree at Tampere University of Technology and is the President's Doctoral School fellow. His research interests include the deployment of knowledge-based and cyber-physical systems in factory automation.

Sari Räsänen
She received MSc degree in Industrial Management and Biotechnology from the Tampere University of Technology, in 2010. Since 2013 she works at the Factory Automation Systems and Technologies Lab of Tampere University of Technology, as the project manager of the eScop project. Before joining the group she worked at project office in R&D department in pharmaceutical industry, and in The Research Institute of the Finnish Economy, researching service process systems and productivity. Her research interests are in management, business and strategies.

Anisha Sampath Kumar
She received BSc degree in Electrical and Electronics Engineering from the Anna University, Coimbatore, Tamil Nadu, India, in 2013. She is currently working towards MSc degree in Factory Automation at the Tampere University of Technology. Since 2014 she works as a research assistant at the Factory Automation Systems and Technologies Lab, Tampere University of Technology. Her research interests are in application of knowledge-bases and cyber-physical systems in factory automation.

Ondřej Severa
He is junior researcher at NTIS (New Technologies for the Information Society) European Center of Excellence at the Faculty of Applied Sciences, University of West Bohemia, Plzeň, Czech Republic. He is also PhD student at the Department of Cybernetics therein. His research interests include real-time control and simulation, embedded systems and human-machine interfaces. He is highly motivated developer with a keen interest in modern web technologies.

Roman Stryjski

He is heading the Institute of Computer Science and Production Management at University of Zielona Góra, Poland. He obtained his MSc (1976) and PhD in Computer Science (1983) from Technical University of Ilmenau, Germany. In 1989 he received DSc (Hab.) at Marin-Luther University Halle/Wittenberg, Germany. His research interests are in the areas of software engineering, systems analysis and design and decision support systems. He is also a member of advisory councils and projects, working as consultant for several companies.

Stanisław Strzelczak

He is Ass. Dean for Int'l Education at Faculty of Production Engineering at Warsaw University of Technology and Guest Professor at North-West University in Xi'an (PRC). His recent research is focused on dynamic behavior of distributed manufacturing. He authored and contributed to 12 books published worldwide, and to ca. 100 papers in journals and proceedings. He teaches Design and Analysis of Manufacturing Systems and Global Operation Strategies. He also runs many PhD seminars at different universities worldwide. He used to work as advisor or non-executive director to corporations, mostly in the aviation and automotive sectors.

Milan Štětina

He is researcher and software developer at the University of West Bohemia in Plzeň, Czech Republic. He obtained MSc degree in Cybernetics therein. He started his working as control engineer in small automation integrator company. He has worked on intrusion prevention system later in his career. He has also gained experience in Enterprise Resource Planning as consultant and software developer. His main research areas are: process control, industrial communication and motion control.

Robert Wojtachnik

He is Ass. Professor at Faculty of Production Engineering, Warsaw University of Technology. His research interests include: performance management systems, strategic management, controlling, business model engineering, web systems design. He teaches courses on MRP databases, programming computer networks, management of innovations, software engineering. He is also Managing Director of Benson Consultants, working for e.g. Orlen PetroTank, Hellmann Worldwide Logistics Poland, DHL Express Poland. He likes to travel, the Argentine tango and photography.

Xiangbin Xu

He received BSc degree in Automation from the Northeastern University, Shenyang, China, in 2008, and the MSc degree (with distinction) in Factory Automation from the Tampere University of Technology, Tampere, Finland, in 2014. Since 2013 he works as project researcher at the Factory Automation Systems and Technologies Laboratory, Tampere University of Technology. His research interests are focused on the areas of cyber-physical systems, manufacturing systems and industrial informatics.