

Ontology-based modeling and configuration of construction processes using process patterns

Ontologie-basierte Modellierung und Konfiguration der
Bauprozesse mit Hilfe von Prozessvorlagen

Alexander Benevolenskiy

Berichte des Instituts für Bauinformatik, Heft 12

Schriftenreihe des Instituts für Bauinformatik
Herausgeber: Univ.-Prof. Dr.-Ing. R. J. Scherer

© Institut für Bauinformatik,
Fakultät Bauingenieurwesen, TU Dresden, 2015

1. Auflage, Februar 2016
ISBN: 978-3-86780-477-6

Institut für Bauinformatik, TU Dresden

Postanschrift

Technische Universität Dresden
01062 Dresden

Besucheranschrift

Nürnberger Str. 31a
2. OG, Raum Nr. 204
01187 Dresden

Tel.: +49 351/463-32966
Fax: +49 351/463-33975
E-Mail: Raimar.Scherer@tu-dresden.de
www: <http://tu-dresden.de/biw/cib>

Diese Arbeit wurde unter dem Titel

Ontology-based modeling and configuration of construction processes using process patterns

Ontologie-basierte Modellierung und Konfiguration der Bauprozesse mit Hilfe von Prozessvorlagen

an der Fakultät Bauingenieurwesen der Technischen Universität Dresden als

DISSERTATION

von

M.Sc. Alexander Benevolenskiy

geboren am 26. März 1985 in St. Petersburg, Russland

zur Erlangung des akademischen Grades eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigt.

Gutachter:

Prof. Dr.-Ing. Raimar J. Scherer, Technische Universität Dresden

Prof. Dr.-Ing. Wolfgang Huhnt Technische Universität Berlin

Tag der öffentlichen Verteidigung: 18. Dezember 2015

Acknowledgement

This work was completed during my research work at the Institute of Construction Informatics of the Technische Universität Dresden from October 2009 to March 2015.

First of all I would like to express my gratitude to my advisor Prof. Dr.-Ing. Raimar Scherer for his support and scientific guidance during these years. That was a very good experience and I did learn a lot during my employment at the Institute of Construction Informatics.

I would like to thank Prof. Dr.-Ing. Wolfgang Huhnt from the Technische Universität Berlin for his role of the second reviewer of this research work and for his time invested in reading and reviewing this thesis.

Moreover I would like to thank all my colleagues at the institute of the Construction Informatics for their assistance and many fruitful discussions in the office or during lunch breaks. My special thanks go to Dr. Ing. Peter Katranuschkov, for his support, especially at the beginning stage of my research work and to Ksenia Roos, with whom we worked closely and productively together on many research topics.

Additionally, I want to thank my friends and especially my best friend Max and his wife Anna for all their support and help at different stages of writing of this thesis.

Finally, I am very thankful to my whole family for their love, encouragement and their trust in me all these years, therefore I would like to dedicate this work to them.

Dresden, December 2015

Alexander Benevolenskiy

Abstract

Process modeling is used in construction to plan and manage the construction process and to support various simulation tasks. A major problem is that due to the one-of-a-kind character of construction projects a lot of work is spent manually developing an overall process schedule for each project. However, the total individual process is typically structured into multiple stages containing a number of recurring similar, but unequal, subprocesses that can be standardized, if appropriately generalized, to generic reusable process patterns. Moreover, not only processes, but also many general construction methods and strategies can be standardized and stored in the form of patterns and configuration rules, which will improve the consistency of modeling and also improve modeling time.

The presented work addresses these issues and presents a new approach for the ontology-based process modeling and its combination with the rule-based process configuration. The proposed system supports the generation of process workflows for construction projects that could be later used in discrete-event simulation software or workflow programs.

A formal high-level model for construction processes and a methodology for using process patterns in the configuration of complex construction tasks are the main focuses of this work. The base idea of the proposed approach is the development and use of two separate, but interrelated, ontologies and their integration with a general-purpose rule-engine. The first ontology is the Process Pattern Ontology, which is used to store reusable process patterns. The second is the Process Instance Ontology, which has similar taxonomy to the Process Pattern Ontology but is uniquely populated with specific process assertions for each construction case.

The developed approach also suggests the application of the process patterns for the configuration of construction processes. This includes the mechanism of the process pattern retrieval, describing the extraction of the required process pattern from the Process Pattern Ontology, the intermediate adaptation step and the configuration step. The configuration step focuses on the integration of the rule-engine with the ontological knowledge-base, as well as on the application of different configuration strategies. With the help of these configuration strategies, realised by means of hierarchical rule sets, first a variation of the process and second a solution for the quick process configuration can be found.

As a practical implementation of the proposed methodology a software prototype called Process Configurator is implemented within this work. This prototype realizes the interaction between all components of the ontology-based configuration approach presented in this work and supports the generation of process schedules for construction projects with the help of reusable process patterns and configuration rules.

Kurzfassung

Prozessmodellierung wird im Bauwesen für die Planung des Bauablaufes genutzt, sowie insbesondere für die Unterstützung unterschiedlicher Simulationsprozesse. Das Hauptproblem ist, dass man wegen des Unikatcharakters der Bauprojekte viel Aufwand und Zeit braucht, um jedes Mal einen Prozessablaufplan des Projektes manuell zu erstellen. Allerdings besteht ein einzelner Gesamtprozess normalerweise aus mehreren Teilen, die ähnliche, aber nicht gleiche Subprozesse beinhalten. Diese Subprozesse können standardisiert, abstrahiert und dann als wiederverwendbare Referenzprozesse generalisiert werden. Außerdem lassen sich nicht nur Prozesse, sondern auch viele Baumethoden und Baustrategien in Form von Konfigurationsregeln formalisieren und speichern. Dies kann die Widerspruchsfreiheit der Modellierung gewährleisten und hat auch das Potential, die Modellierungszeit zu reduzieren.

Die vorliegende Arbeit zielt darauf ab, diese Aspekte zu analysieren und einen neuen Ansatz zur Kombination der ontologiebasierten Prozessmodellierung mit der regelbasierten Prozesskonfiguration vorzustellen. Das dargestellte System unterstützt die Erstellung der Prozessablaufpläne für die Bauprojekte, die danach mit Hilfe von Simulationssoftware simuliert werden.

In dieser Dissertation werden die Entwicklung eines formalen Modells für Bauprozesse und die Methodologie der Nutzung der Referenzprozesse bei der Konfigurierung komplexer Bauaufgaben beschrieben. Die grundlegende Idee des vorgeschlagenen Ansatzes ist die Entwicklung und die Verwendung zweier Ontologien. Die erste Ontologie wird als Process Pattern Ontologie bezeichnet und zur Speicherung der wiederverwendbaren Referenzprozesse eingesetzt. Die zweite Ontologie mit der Bezeichnung Process Instance Ontologie speichert die Prozessinstanzen für die spezifischen Bauprojekte. Die beiden Ontologien haben ähnliche, aber nicht gleiche, Strukturen und sind über eine Regel-Engine integriert.

Der entwickelte Ansatz beinhaltet ebenfalls die Nutzung der Referenzprozesse im Konfigurationsprozess. Dies umfasst sowohl einen Mechanismus zur Abfrage der Referenzprozesse, welcher das Extrahieren des benötigten Referenzprozesses aus der Ontologie beschreibt, als auch die Anpassung der Referenzprozesse sowie deren Konfiguration mit Regeln. In der Konfiguration liegt der Fokus auf der Integration der Regel-Engine mit der ontologischen Wissensbasis und die Anwendung unterschiedlicher Konfigurationsstrategien. Mittels Konfigurationsstrategien, die mit Hilfe von hierarchischen Regelmengen realisiert werden, kann eine intelligente Lösung für die schnelle Prozesskonfiguration gefunden werden.

Als praktische Implementierung der vorgeschlagenen Methodologie wird ein Prototyp, als Process Configurator bezeichnet, entwickelt. Der Process Configurator realisiert die Interaktion zwischen allen Komponenten des Systems und unterstützt die Erstellung der Prozessablaufpläne mit Hilfe der Referenzprozesse und Konfigurationsregel.

Table of Contents

| | |
|---|-----------|
| 1. Introduction | 13 |
| 1.1. Motivation | 13 |
| 1.2. Problem statement | 14 |
| 1.3. Hypotheses | 15 |
| 1.4. Objectives | 15 |
| 1.5. A new approach of process modeling and configuration | 17 |
| 1.6. Thesis outline | 20 |
| 2. Process modeling | 22 |
| 2.1. Basic issues | 22 |
| 2.2. Business process modeling | 25 |
| 2.3. Process modeling techniques | 28 |
| 2.3.1. Integration Definition IDEF | 28 |
| 2.3.2. UML | 30 |
| 2.3.3. EPC | 32 |
| 2.3.4. BPMN | 35 |
| 2.3.5. Traditional modeling techniques | 37 |
| 2.4. Process modeling in construction industry | 41 |
| 2.4.1. Main features of process modeling in construction industry | 41 |
| 2.4.2. Existing approaches in construction industry | 44 |
| 3. Ontology-based modeling | 58 |
| 3.1. Knowledge-based models | 58 |
| 3.2. Ontology | 64 |
| 3.2.1. Ontology description languages | 65 |
| 4. Modeling of construction processes and process patterns | 78 |
| 4.1. Reference modeling | 78 |
| 4.2. Process pattern | 81 |
| 4.2.1. Examples | 83 |
| 4.3. Ontological framework | 88 |
| 4.3.1. Motivation for using ontologies instead of databases | 88 |

| | | |
|------------|--|------------|
| 4.3.2. | Ontology for process patterns | 93 |
| 4.3.3. | Ontology for process instances | 99 |
| 5. | Configuration of construction processes | 103 |
| 5.1. | Generation and configuration | 103 |
| 5.2. | Process pattern retrieval | 104 |
| 5.3. | Process adaptation | 107 |
| 5.4. | Process configuration | 109 |
| 5.4.1. | Configuration rules | 109 |
| 5.4.2. | Homogenous approach | 111 |
| 5.4.3. | Hybrid approach | 115 |
| 5.4.4. | Types of rules | 119 |
| 5.4.5. | Configuration strategies | 122 |
| 6. | Implementation | 128 |
| 6.1. | System architecture and its components | 128 |
| 6.2. | Process configurator | 130 |
| 6.2.1. | BPMN mapping | 136 |
| 7. | Case study | 140 |
| 7.1. | Case 1 | 140 |
| 7.2. | Case 2 | 146 |
| 8. | Conclusions and future research | 149 |
| 8.1. | Conclusions | 149 |
| 8.2. | Outlook | 150 |
| 9. | Appendix A. | 156 |
| A.1. | Process Pattern | 156 |
| A.2. | Rules overview | 158 |
| A.3. | Rules implementation | 160 |
| 10. | References | 163 |

List of Abbreviations

| | |
|---------|---|
| ABox | Assertion Box |
| API | Application Programming Interface |
| BIM | Building Information Model |
| BPO | Business Process Object |
| BPM | Business Process Modeling |
| BPMN | Business Process Modeling Notation |
| CPM | Critical Path Method |
| CWA | Closed World Assumption |
| DMS | Defect Management System |
| DL | Description Logic |
| EPC | Event-driven Process Chain |
| eEPC | Extended Event-driven Process Chain |
| ERM | Entity Relationship Model |
| F-Logic | Frame Logic |
| GC | General Contractor |
| GEPM | Generic Construction Process Modeling |
| GUI | Graphical User Interface |
| IDEF | Integration Definition |
| IDEF0 | Integrated Definition for Function Modeling |
| IFC | Industry Foundation Classes |
| KB | Knowledge Base |
| KIF | Knowledge Interchange Format |

| | |
|--------|---|
| LHS | Left Hand Side |
| LoD | Level of Details |
| OMG | Object Management Group |
| OWA | Open World Assumption |
| OWL | Web Ontology Language |
| PERT | Program Evaluation and Review Technique |
| PIO | Process Instance Ontology |
| PPO | Process Pattern Ontology |
| RDF | Resource Description Framework |
| RDFS | Resource Description Framework Schema |
| RHS | Right Hand Side |
| SPARQL | SPARQL Protocol and RDF Query Language |
| SQL | Structured Query Language |
| SWRL | Semantic Web Rule Language |
| TBox | Terminological Box |
| UML | Unified Modeling Language |
| URI | Universal Resource Identifier |
| WfMC | Workflow Management Coalition |
| W3C | World Wide Web Consortium |
| XML | Extensible Markup Language |
| XTM | XML Topic Maps |

1. Introduction

This chapter provides an introduction to this doctoral thesis. After a discussion of general motivation in the first section, we continue with objectives of this work in the second section. Finally, the last section closes this chapter by presenting a thesis outline.

1.1. Motivation

Modeling plays a significant role in representing and describing complex construction processes at a more abstract level. In the construction industry, process modeling and configuration are used more often to support simulation, to estimate and to plan required resources and costs. Due to unique¹ character of construction projects the manual development of a project overall process schedule is a very time-consuming procedure and hence it is very seldom carried out without a support of any software tools. However, the total individual process usually contains a number of repetitive similar subprocesses.

Such subprocesses can be represented and stored as generic reusable process patterns that can be standardized and very individually instantiated for many different projects and processes, and consequently, process modeling and scheduling time can be considerably reduced.

In many areas like information technology, government, finances and insurance, process modeling became an essential factor for companies to stay competitive and optimize their work. Therefore they use different process modeling techniques to capture and standardize their processes and to specify their work and resources. Moreover, some typical operations in these companies are described in the form of generic reference processes that can be used as reusable solutions for the development of new models. The use of these reference process models can reduce costs and time and improve the quality of the processes, because reference process models represent normally proven solutions from the best practices. Furthermore, the variety of different IT technologies and tools supporting process and reference process modeling can speed up and simplify the development process. At the same time in the construction sector a lot of traditional methods, which usually do not have an adequate IT support, are still used.

The motivation of this thesis is to analyze the area of the process and reference process modeling in general and for the construction industry in particular and to provide a meth-

¹ For convenience the word “unique” is used throughout this dissertation instead of the long expression “one-of-a-kind”

odology for the ontology-based modeling of construction processes and their configuration. In comparison with databases an ontological model is more suitable to provide a flexible structure for modeling construction projects, as it can be easily changed, adopted and integrated with a rule engine.

In this work the main concepts of an approach that allows interactive configuring of the construction process, depending on the actual availability of resources, pre-given construction techniques and individual user requirements are presented.

1.2. Problem statement

As discussed in the previous section, because of the unique character of construction projects, many participants and domain models involved, the design of the overall construction process model is a very time-consuming task. However the importance of this construction process model rises significantly in the last years, because this model can be very useful not only for generating of process workflows, simulation or cost estimation (Scherer & Schapke, 2014), but also some concepts of it and the proposed methodology can be applied for many other different tasks, e.g. the support of the interoperability between energy analysis tools and product and building design tools (ISES, 2014) (Kadolsky, et al., 2014) or the design of energy-efficient constructions and their optimal energetic embedding in the neighborhood of surrounding buildings and energy systems (eeEmbedded, 2015).

The challenges that this research work addresses can be formulated in the following research questions:

How should the construction processes be formalized and modeled in such a way, that they provide a general and extendable structure for describing typical construction tasks with their resources, objects and subprocesses?

How can we define a knowledge base for modeling and storing of reusable construction process patterns?

How a collaborative process model with clear and unambiguous semantics can be defined, that can simplify the planning and management of construction processes and make them transparent for all involved participants?

How can an approach be for the configuration of construction processes at different process levels and how can it be realized with the use of a general purpose rule language?

1.3. Hypotheses

1. The ontological process model used for the formalization of processes can be easily extended and interconnected with other domain models and therefore it can simplify the planning and management of construction processes and make them transparent for all involved participants.
2. The ontological knowledge base representing a set of typical construction process patterns with related construction data can provide a collaboration model with clear and unambiguous semantics, thereby facilitating integration and reuse of existing best practice business processes in the form of patterns and data models.
3. The use of the ontological structure for the formalization of the construction process model provides possibilities to apply reasoning mechanism and that can improve the overall quality of the model.
4. The ontological process model in combination with the general purpose rule engine can be used to define different kinds of configuration rules and strategies that can modify and improve the construction process.
5. An automated procedure realized by means of ontologies and enhanced with additional configuration mechanisms can be very effective, enabling the combination of adapted process patterns set into a consistent process chain, which can be used for further applications and optimization algorithms.

1.4. Objectives

The research objectives of this thesis are to develop a high-level process model for the construction processes based on ontology and to define the reusable process patterns based on this model, which is much more flexible, extensible, generic and interoperable than existing object-oriented and relational database models. Furthermore, some aspects of the configuration and verification of the construction processes are considered.

In order to reach the research objectives of this thesis following steps have to be done:

- The current state of research in the area of process modeling has to be analyzed. In particular focus should be the concept of the Business Process Modeling that has become a main methodology for different engineering fields.
- Before introducing an ontological model for the construction processes, it is necessary not only to describe the theoretical background of this knowledge representation formalism, but also to take a look at some successful implementations of ontologies for modeling processes. Special review requires the meth-

od of Description Logic, a family of knowledge representation languages, providing a logical formalism for ontologies.

- An ontological model for the construction processes has to be developed. That includes not only the definition of the hierarchy of concepts, but also modeling a property set of these concepts and all required relations between concepts in the ontology.
- Based on this ontological model, an ontological framework has to be developed. It includes two interrelated ontologies based on the same schema, but further extended to two different specializations. The first ontology should be used to model and store reusable process patterns and the second one is used for the specific construction process descriptions. In addition, a particular focus will be on the definition and implementation of the concept of the process pattern, that should be generic, but at the same time specific enough in order to be useful for the process configuration.
- A methodology for the configuration of construction processes has to be proposed and implemented. It includes wide variety of configuration aspects, starting from the retrieval of the process pattern and finishing with finding the optimal sequence of processes. Therefore, it is also necessary to define and classify different aspects of the configuration process.
- A software-prototype for the verification of the developed concepts has to be implemented. It is also necessary to show with the help of several use cases, how the proposed solution could support and speed-up the planning process

1.5. A new approach of process modeling and configuration

In this section a new approach for process modeling and configuration based on ontology and rules that was developed in this work is introduced shortly.

Most of the other proposed nowadays process configuration methodologies use databases to store their process models and are based on planning and scheduling (Benevolenskiy, et al., 2012). In (Fischer M., Aalami F., 1995) a knowledge-based environment, integrating construction methods, product models and information resources with construction plans and schedules, where as an input for scheduling a 3D CAD model is used, is introduced.

The reuse of experience from former projects and the generation of construction schedules with the case-based reasoning support are investigated in (Tauscher, et al., 2007). In that work Feature Logic (Pahl & Damrath, 2001) is used to model construction tasks and constraints as well as to support the generation of workflow graphs. An addition combination of this approach with the 4D animation methodology is presented in (Tauscher, 2011).

In the concept of 4D CAD construction processes are connected with respective components of the 3D CAD model. In (Huhnt, Richter, Wallner, Habashi, & Krämer, 2010) the description of process patterns and their structure in such 4D-systems is presented in detail. However, it is necessary to say, that 4D is often used for visualization and not for the process configuration and optimization.

The novelty of the proposed approach is that an ontology based-model for construction processes uses reusable process patterns for the process instantiation and is integrated with a configuration knowledge base used for the configuration and the optimization of processes. The approach consists of two main stages, which are presented at the Figures below:

- Stage one: Instantiation and configuration with patterns (s. Figure 1-1)
- Stage two: Configuration with rules (s. Figure 1-2)

An ontological model is used to store process model together with relations to other domain models as well as generic reusable process patterns. The advantage of the approach is that the usage of the ontology allows constructing a flexible and extendable high-level model for processes and combining it with the configuration rules. Moreover, this model has an extendable schema and therefore references to new domain models can be easily added to it. The description of this ontological model, its main concepts and the ontological framework can be found in chapter 4. The configuration process as well as search and the adaptation mechanisms are described in chapter 5.

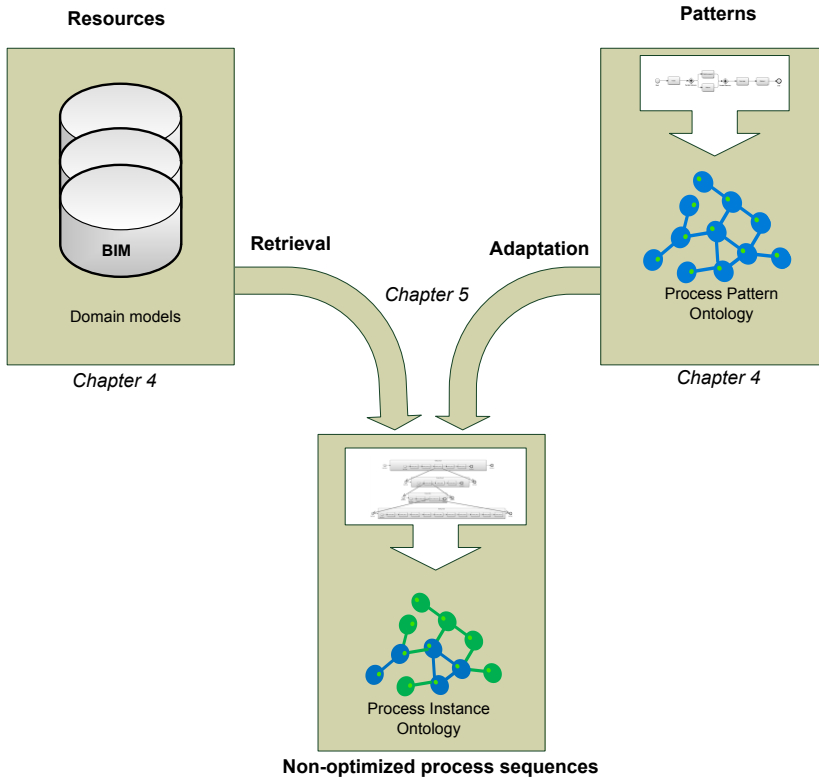


Figure 1-1: Stage one: retrieve, adopt and link

Figure 1-1 shows how from the information stored in different domain models and the generic process patterns a new process model can be generated. This process model contains process descriptions, relations to other models and process sequences, which can be further configured and optimized at the stage two.

The stage two is an additional step because the process model already contains all process sequences after the stage one. However some processes can still be further configured or optimized with the help of configuration rules and strategies. For that purpose an ontological framework described in chapter 4 is integrated together with a rule engine. The description of the technical implementation of the configuration process as well as the overview of the configuration rules can be found in chapter 5.

Figure 1-2 shows how the initial process model (shown to the left) together with the configuration knowledge base (shown to the right) can be used to generate the optimized process model.

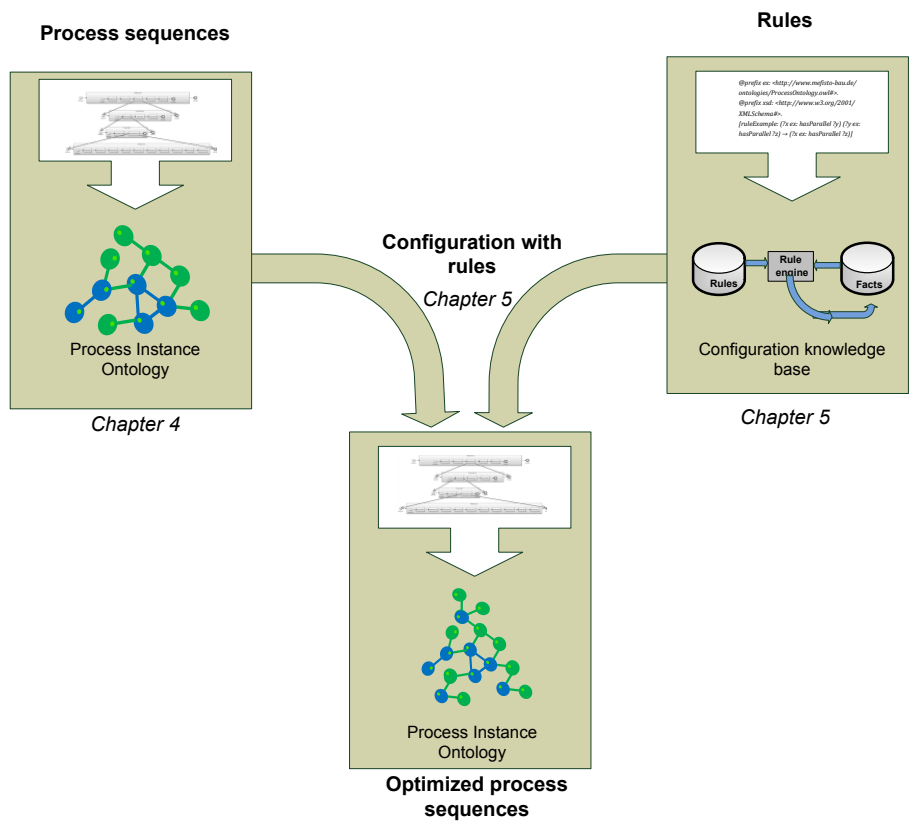


Figure 1-2: Stage two: order and optimize

In this thesis the basics and the basis approach for a knowledge based optimization are shown, however it was not the main focus of this research work. It should be mentioned here, that the optimization is not restricted to the knowledge application by rules. Knowledge can be as well applied through algorithms where several approaches exist, as shortly shown and summarized in chapter 2.

This thesis had not the objective to optimize processes, but to configure them with a high degree of formalization and present processes in such a way, that the rule-based knowledge application for the optimization can be applied.

Some limitations of the proposed approach also should be mentioned here. The complexity of the ontology engineering process and the implementation of rules as well as a limited scalability and robustness of the ontological data model in comparison with the relational data model represent some of the bottlenecks of the approach. More detailed descriptions and ways to overcome them are provided in chapter 4 and chapter 6.

1.6. Thesis outline

This thesis is organized in eight chapters. Chapter 1 introduces research and includes the motivation and objectives of this work. In chapter 2 the state of the art of the process modeling is presented and discussed. Chapter 3 reviews an ontology-based modeling approach and focuses on the theoretical background of this approach. In chapter 4 an ontological model for the construction processes is presented and a concept of a process pattern is introduced. Different aspects of the configuration of construction processes are described in the chapter 5. Finally, in chapters 6 and 7 a prototypical implementation and two case studies are presented. Chapter 8 closes this thesis by presenting the conclusions of the thesis and providing a recommendation for future work.

Chapter 1: Introduction

In this chapter the motivation and objectives of this thesis are presented.

Chapter 2: Process modeling

This chapter reviews the background of the process modeling. In particular, we focus on business process modeling and discuss some process modeling techniques. The state of the art of process modeling in construction is introduced.

Chapter 3: Ontology-based modeling

This chapter gives a short overview of the theoretical background of the ontology-based modeling. The basic concepts of description logic and an overview of the most common ontology description languages are presented. Furthermore, in this chapter some of ontologies for processes are reviewed.

Chapter 4: Modeling of construction processes and process patterns

In this chapter a concept of the process pattern is introduced. Since the technological implementation of this concept is based on the ontology, an ontological framework, consisting of two inter-related, but independent ontologies, one for the process pattern and another for the process instances, is proposed and described in this chapter.

Chapter 5: Configuration of construction processes

This chapter presents and discusses different aspects of the generation and configuration of construction processes. At first the pattern retrieval and adaptation process is explained. For that purpose a specific query language for the ontological models as well as the query mechanism are discussed briefly. After that different configuration ap-

proaches are presented. Finally, the concept of the configuration strategy and its realization will be introduced.

Chapter 6: Prototypical implementation

In this chapter a prototypical implementation of the developed approach is presented. At first a general architecture of the system including all its components and their interaction is outlined. Finally, a java-based program prototype, which is developed in this work, is presented.

Chapter 7: Case study

This chapter presents two case studies in order to validate and prove the developed concepts. Both cases are based on existing project data of a real-world construction projects. The first case study is based on a model of a high-rise office building. In the second case study a five-storey school building is considered.

Chapter 8: Conclusions and future research

This chapter gives a summary of this thesis and offers an outlook for future research. In particular, various possibilities to extend the presented methodology are discussed.

2. Process modeling

This chapter provides an overview of the basic concepts of the process modeling domain. In the first section different purposes of the process models and their classification is discussed. This is followed by a presentation of the business process modeling approach in section two. Three different types of business processes: core, support and management processes, are introduced there. In this section possible purposes of the Business Process Modeling are also mentioned.

In section three different process modeling techniques that are commonly used currently are presented, including some business-oriented approaches, such as BPMN or EPC, as well as traditional modeling techniques like Petri-nets or PERT.

Finally, in the last section main features and applications of the process modeling in the construction industry are considered and the detailed overview of some existing approaches is given.

2.1. Basic issues

Nowadays the term process modeling is used in various fields and its exact definition usually depends on the context of using. Process modeling is rather a new research area, which is often considered as a subpart of the process engineering area. Business process modeling (BPM) plays an important role in the process modeling. Its main concepts, methods and techniques are described in the section 2.2. In this section some definitions and classification of the process model in general are given.

Process model is a main concept in process modeling. A good definition of a process model is given in (Rolland, 1998):

“A process model describes the common properties of a class of processes having the same nature.” Therefore it could be said, that process model describes processes on the so called *type level*.

Process model can serve different purposes. In his work (Rolland, 1998) the author summarizes 3 main purposes of process models: *descriptive, prescriptive and explanatory*:

- A *descriptive purpose* of a process model consists of process description itself and determination of possible improvements that can make the performance more effective or efficient.

- A *prescriptive purpose* defines the desired processes and the way how they should be performed. This can be done with the help of different rules, guidelines or behavior patterns.
- An *explanatory purpose* is particularly important for processes, where several courses of action are possible. It explores and evaluates them basing on rational arguments.

Three groups were proposed for the classification of the process models in (Rolland, et al., 1999):

Activity-oriented models, which are concentrated on the activities performed for the specific purpose. Frequently used in the software development process the Waterfall model (s. Figure 2-1), that was firstly described formally in (Royce, 1970) is a good example of the activity-oriented process model. Other examples of activities-oriented models are the Spiral model (s. Figure 2-2) (Boehm, 1988) and the Fountain model (s. Figure 2-1) (Henderson-Sellers & Edwards, 1990) from the software engineering.

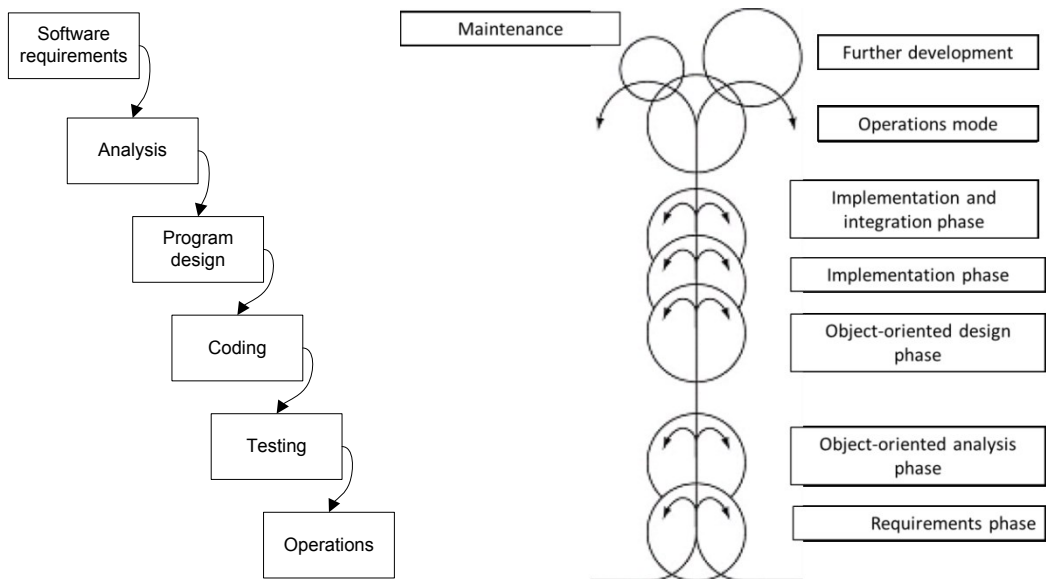


Figure 2-1: The Waterfall (left) and the Fountain (right) models

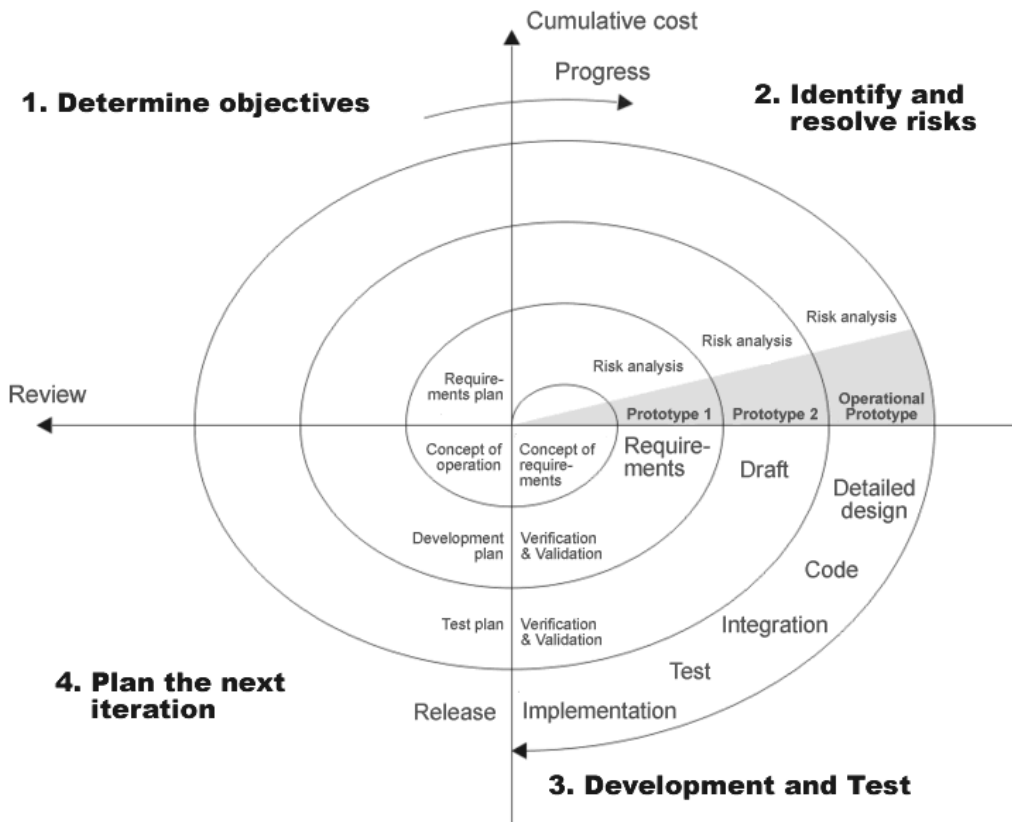


Figure 2-2: The Spiral model

Product-oriented models are defined similar to the activity-oriented models, but activities are linked additionally to their outputs. Therefore, product-oriented models define the process through the transformation of the product. Three concepts composing the base of the meta-model are: *activity*, *product state* and *state transition*. The Viewpoints model (Filkenstein, et al., 1990) and the Entity Process Model (Humphrey, 1989) from the software engineering are good examples of the product-oriented models.

Decision-oriented models are focused not only on the activities performed and the product, but also on the set of decisions behind this transformation. The decision-oriented models can explain both, the way how the process takes places and why it takes places. Therefore, these models are semantically more powerful than product-oriented ones. The Generic Model for Representing Design Methods described in (Potts, 1989) and based on the Issue-Based Information System (IBIS) approach belongs to the category of decision-oriented process models. For example the Generic Model presented in Figure 2-3 (Potts, 1989) is designed to be as general as possible and contains only five enti-

ties types to describe base classes as well as binary relationships between them to describe decisions.

After considering main groups of process model in general we will continue in the next chapter with the more specific approach, which is mostly used nowadays, – a business process modelling approach.

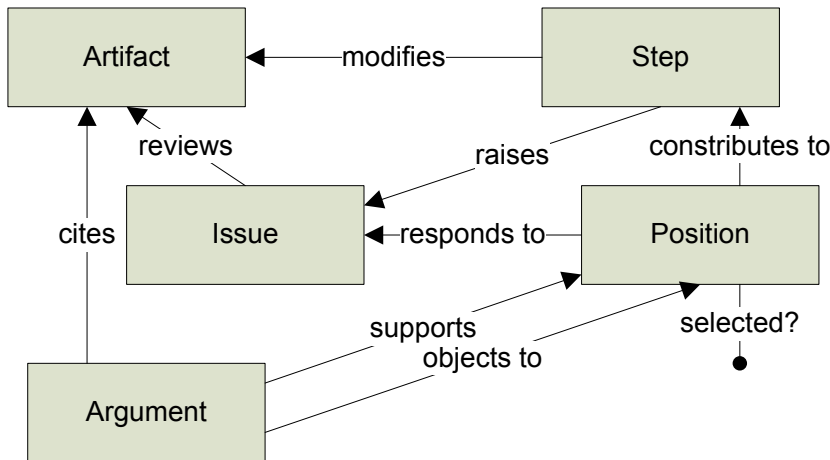


Figure 2-3: Generic Model for Representing Design Methods

2.2. Business process modeling

Over the last decades the term process became a new productivity paradigm. Many companies started “*to think in processes*” and that resulted in the fact, that many enterprises today are focusing on capturing their business processes in order to improve their business performance. Firstly this trend began in the software engineering field, but then due to the rapid development of information technologies it has spread across many other branches.

Business Process Modeling is an activity that has its goal to represent processes of an organization in order to analyze them and improve organizational efficiency and quality. Nowadays the *Business Process Modeling* methodology is being used in nearly all kind of organizations from governmental organizations to academic institutions. However, even that many techniques to model Business Processes such as Gant chart, flowcharts, PERT diagram and many others were already introduced in the middle of the 20th century, the active usage of them in context of the *Business Process Modeling* began only at the end of the 20th century.

Two main notions in *Business Process Modeling* are *Business Process* and *Business Process Model*. The definition of the Business Process is very similar to the definition of the *Process* in general, but it concretizes the last one in context of business activities.

In (Davenport, 1993) Davenport gives the following definition of *Process*, which also can be used for the *Business Process*:

"A process is simply a structured, measured set of activities designed to produce a specific output for a particular customer or market. It implies a strong emphasis on how work is done within an organization, in contrast to a product focus's emphasis on what".

Three types of business processes shown in (Figure 2-4) (Ould M. A., 1995) are:

- Core processes;
- Support processes;
- Management processes.

Core processes (in some other works also called operational processes) concentrate on satisfying external customers. These processes respond to the customer request and generate customer satisfaction. Core processes directly add value to the business.

Support processes have its goal to satisfy internal customers. By supporting the core processes they might add value to the customer indirectly.

Management processes govern the operation of a system in general by managing the core or the support processes.

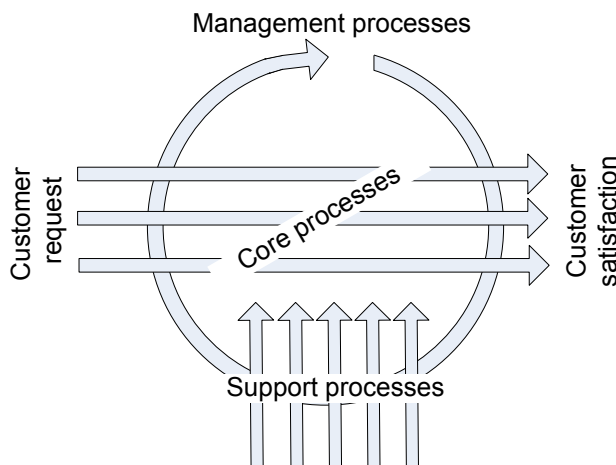


Figure 2-4: The three types of business processes

At this point it is necessary to provide the definition of the term *Workflow*, because this term is often confused with the term *Process* and even that their definitions are quite similar, there are some important differences between them. The Workflow Management Coalition (WfMC) defines the term *Workflow* as (WfMC, 1996):

“The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.”

Therefore we can consider a *Workflow* as a kind of the representation of the Business Process in its execution level, which is used to automate it.

A *Business Process Model* is a structural representation, diagram or description representing the sequence of activities and relations between them. Therefore *Business Process Model* can be seen as the result of a mapping of a *Business Process*.

In some cases two different types of *Business Process Model* are considered. The first type presents a so called “*AS IS*” model, representing the current situation. The second type is a “*TO BE*” model, representing the intended new situation. For example change management program usually has its goal to improve Business Processes in some organization by moving from the “*AS IS*” model to the “*TO BE*” model.

The main aim of *Business Process Modeling* is to capture and illustrate *Business Processes* in a company in order to edit or improve them later. In (Rosenkranz, 2005) following possible purposes of *Business Process Modeling* are mentioned:

- Orientation of Business Process Modeling to customer needs and benefits;
- Documentation, storage and archiving of organizational knowledge in databases(process warehouses);
- Increase transparency of the processes;
- Controlling of the process organization;
- Support by the standardization/certification (ISO);
- Increase of process efficiency through:
 - Process redesign;
 - Process innovation;
 - Process optimization;
- Simulation of the introduction of new technologies and organizational forms.

2.3. Process modeling techniques

Before choosing appropriate process modeling techniques it is necessary to identify the purposes of the models, because different techniques can be suitable for different purposes. For example in his work Phalp (Phalp, 1998) argues that two important considerations in process modeling are notation and method. Different notations have different capabilities and that is the main reason why that should be considered at first. The same process model with two different notations has different properties and can be used for completely different purposes.

There are many business process modeling techniques coming from different areas, such as Unified Modeling Language (UML) (Fowler M., 2004) initially used in software design, Petri nets coming from mathematical theory for process analysis (Petri, 1962) or business-oriented modeling approaches such as Business Process Modeling Notation (BPMN) (White, 2004) and Event-driven Process Chain (EPC) (Scheer, 2000). Various research works provide comprehensive reviews of many of them (Aguilar-Savén, 2004), (Recker, et al., 2009), (Carnaghan, 2006).

Two main categories can be proposed for the existing business process modeling techniques (Recker, et al., 2009). The first one includes intuitive graphical modeling techniques such as EPC, that are mostly concerned with capturing and understanding business processes and discussing requirements and process improvements with subject experts. The modeling techniques from the second category are mostly focused on the process analysis or process execution. They are usually based on a strong mathematical basis and a typical example of them is Petri nets.

In the following sections we consider some of process modeling techniques from the first category that are commonly used nowadays.

2.3.1. Integration Definition IDEF

IDEF (Integration Definition) is a family of modeling languages developed with a support of US Air Force in order to improve manufacturing operations².

The IDEF family consists of many components that are used for different applications:

- IDEF0 : Function modeling
- IDEF1 : Information Modeling
- IDEF1X : Data Modeling
- IDEF2 : Simulation Model Design

² Integration Definition Methods <http://www.idef.com> Retrieved 2013-05-10

- IDEF3 : Process Description Capture
- IDEF4 : Object-Oriented Design
- IDEF5 : Ontology Description Capture
- IDEF6 : Design Rationale Capture
- IDEF7 : Information System Auditing
- IDEF8 : User Interface Modeling
- IDEF9 : Business Constraint Discovery
- IDEF10 : Implementation Architecture Modeling
- IDEF11 : Information Artifact Modeling
- IDEF12 : Organization Modeling
- IDEF13 : Three Schema Mapping Design
- IDEF14 : Network Design

IDEF0 (Integrated Definition for Function Modeling) is one of the most popular tools for functional modeling. Moreover, this is the most useful standard from IDEF family for the business process modeling and therefore it will be explained below.

IDEF0 is defined in (Defense Dept., Defense Acquisition University, 2001) as a common modeling technique for the analysis, development, re-engineering, and integration of information systems, business processes or software engineering analysis. It can be used to represent graphically a wide variety of business, manufacturing and other types of operations at different level of detail. Moreover, IDEF0 is capable to model various automated and non-automated systems. It can specify the functions and define the requirements for the new systems or analyze the function of the existing one.

IDEF0 model consists of a hierarchical series of diagrams, text and glossary, which are cross-referenced to each other. Two main modeling components defined in IDEF0 are:

- Functions (represented graphically as boxes);
- Data and objects (represented graphically as arrows).

A simple IDEF0 box format is shown in Figure 2-5 below (Defense Dept., Defense Acquisition University, 2001).

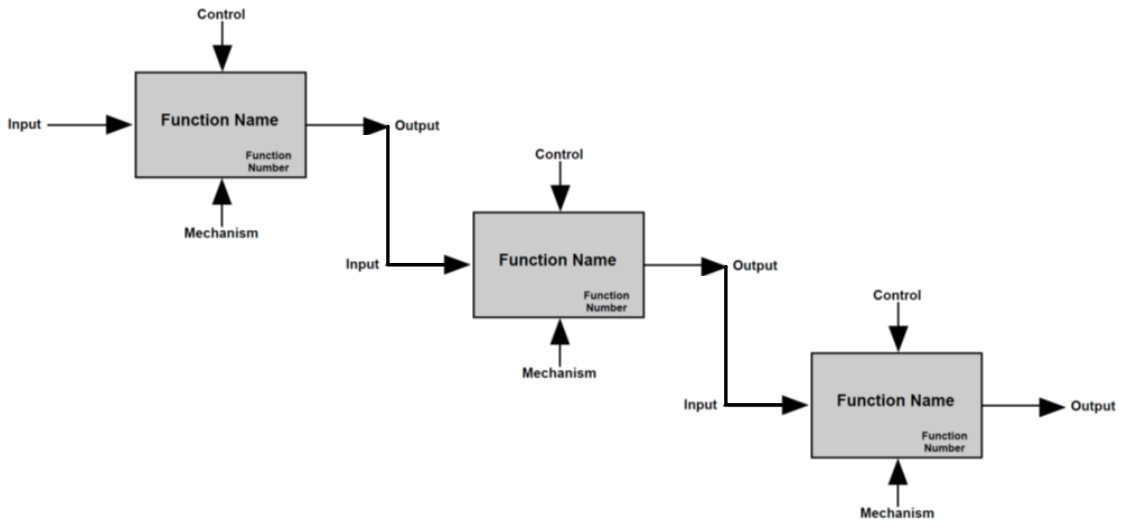


Figure 2-5: Integration Definition for Function Modeling (IDEF0) Box Format

2.3.2. UML

The Unified Modeling Language (UML) is an open standard and one of the most widely used modeling language for the specification, design and documentation of software systems.

It was created by Grady Booch, Ivar Jacobson and Jim Rumbaugh (Booch, et al., 1998) and is maintained by the Object Management Group (OMG)³. UML combines different modeling techniques from data, business, component and object modeling.

UML specification has few predefined diagrams capturing three important aspects of systems: structure, behavior and functionality.

In (Ambler, 2010) three classifications of UML diagrams are mentioned:

- *Behavior diagrams* depict behavioral features of a system or business process. This includes activity, state machine and use case diagrams as well as four interaction diagrams.
- *Interaction diagrams* are a subset of behavior diagrams, which emphasize object interactions. This includes communication, interaction overview, sequence and timing diagrams.

³ <http://www.uml.org> Retrieved 2013-05-14

- *Structure diagrams* depict the elements of a specification that are irrespective of time. This includes class, composite structure, component, deployment, object and package diagrams.

UML is mainly used to model software systems, but it is also can be very suitable for the business modeling. Eriksson and Penker show in (Eriksson & Penker, 1998) the use of UML for business modeling and present a variety of reusable business patterns. Process modeling patterns, presented in (Eriksson & Penker, 1998), are focused on how to model a process in order to achieve a high quality of the model. The Basis Process Pattern structure from (Eriksson & Penker, 1998) is shown in Figure 2-6.

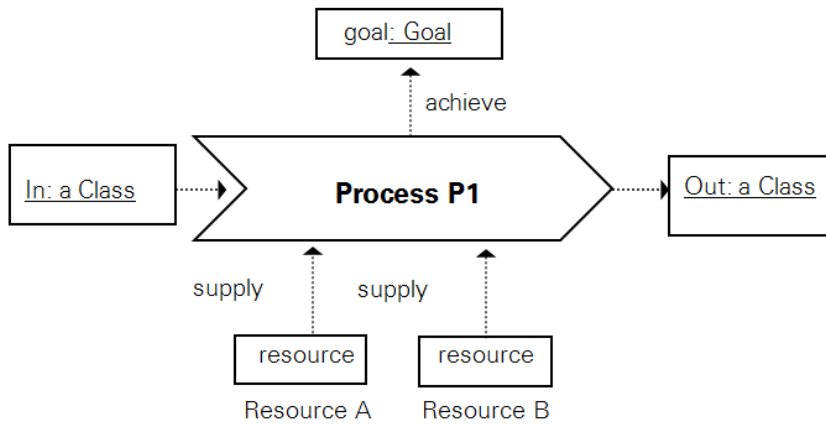


Figure 2-6: The Basic Process Pattern Structure

Comparison with IDEF0

UML as well as IDEF0 can be used for the business process modeling, however in comparison with IDEF, UML has a number of advantages, some of which are summarized in Table 1 from (Wilcox & Gurau, 2003).

Table 1: Comparison between IDEF and UML

| Business modelling languages | Notation | Standardisation | Portability |
|------------------------------|------------------|-------------------------------|---------------------------|
| IDEF | Complex, rigid | Low level of standardisation | Low level of portability |
| UML | Simple, flexible | High level of standardisation | High level of portability |

2.3.3. EPC

An Event-driven Process Chain (EPC) was introduced by Keller, Nüttgens and Scheer (Keller, et al., 1992) in 1992. It is a method to model business processes that was developed within the framework of Architecture of Integrated Information Systems, ARIS. EPC is widely used by many companies in their software applications in order to describe processes at the level of their business logic.

EPC diagram contains different elements. The four basic elements are presented in Figure 2-7.





| | |
|---|-------------------|
|  | Function |
|  | Event |
|  | Logical Connector |
|  | Control Flow |

Figure 2-7: The four basic EPC elements

- *Function*: Function is the basic building block in EPC, which represents tasks or activities. Functions in EPC are active elements, because they describe the transformation of element from one state to another. Functions are named with a verb and are represented graphically as rounded boxes.
- *Event*: Events are passive elements in EPC describing incidence of a state. They usually link function with each other and represent their pre- and post-condition. Graphical representation of an event in the EPC graph is a hexagon. Generally each process in EPC should begin and end with an event.
- *Logical Connector*: Logical connectors describe the logical relationship between the elements in EPC. To represent three different logical operations three kinds of connectors are used in EPC (AND, XOR, OR).
- *Control Flow*: Control flows describe the chronological dependency of event and functions and connect events, functions and logical connectors with each other. An arrow is a graphical representation of a control flow in the EPC graph.

An example of the EPC diagram describing a construction of a column is shown in Figure 2-8.

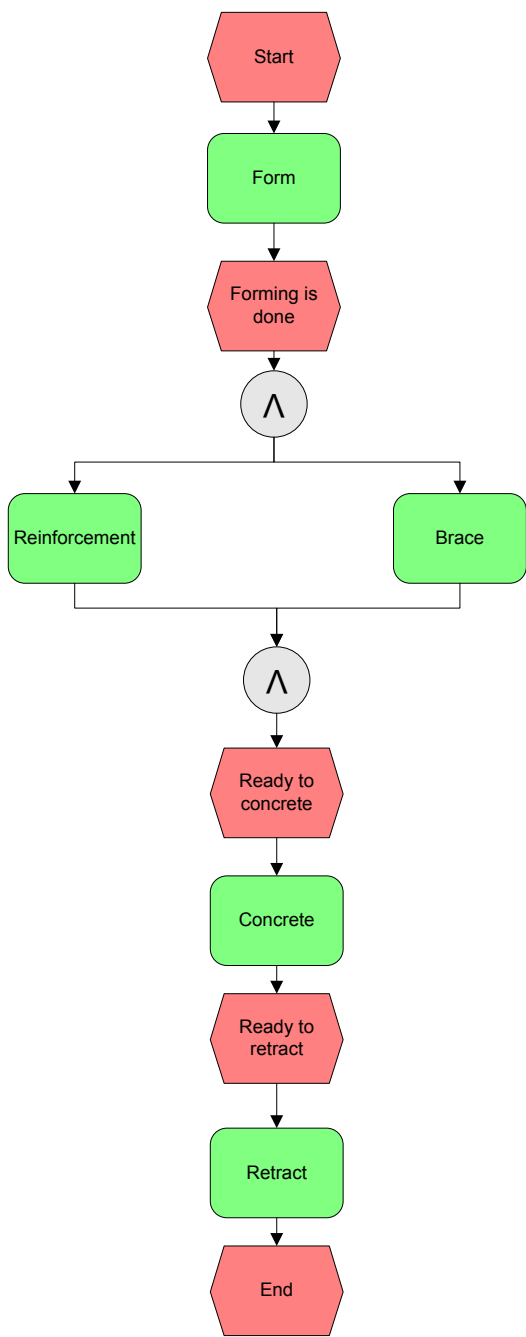


Figure 2-8: An example of a simple EPC diagram describing the basic process model for producing a RC element

2.3.4. BPMN

Business Process Model and Notation (BPMN) is a standard developed and maintained by the Object Management Group providing a notation for specifying business processes⁴. It has some similarities with the activity diagram from UML and has its goal to provide such a notation for the business process modeling that is easily understandable by business users, but is able to represent complex process semantics too.

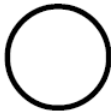

BPMN has also executable capabilities, because it provides the mapping mechanism for transformation of models into an executable form (for example by using a Business Process Execution Language).

The five basic categories of graphical elements in BPMN are: Flow Objects (Events, Activities and Gateways), Data (Data Objects, Data Inputs, Data Outputs and Data Stores), Connecting Objects (Sequence Flows, Message Flows, Associations and Data Associations), Swimlanes (Pools and Lines) and Artifacts (Group and Text Annotation).

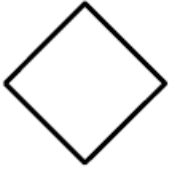


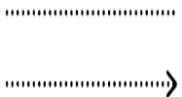




Flow objects (Events, Activities and Gateways) represent the main graphical elements defining behavior and structure of business processes. Table 2 (after Business Process Model and Notation, v2.0, 2011) below gives a description and the notation of some basic modeling elements in BPMN.

An example of BPMN diagram for a simple construction process is shown in Figure 2-9.

Table 2: Basic modeling elements in BPMN

| Element | Description | Notation |
|----------|---|---|
| Event | An Event is something that “happens” during the course of a Process. Events are represented with a circle and can be of three types: Start, Intermediate and End. |  |
| Activity | An Activity is a generic term for work that must be done. The two types of Activities that are part of a Process Model are: Sub-Process and Tasks, which are represented as rounded rectangles. |  |

⁴ <http://www.omg.org/spec/BPMN/> Retrieved 201-02-05

| | | |
|----------------------|--|---|
| Gateway | A Gateway is used to control the divergence and convergence of Sequence Flows in a Process. It is represented with a diamond shape. Internal markers indicate the type of behavior control (branching, forking, merging and joining of paths). |  |
| Sequence Flow | A Sequence Flow is used to show the order that Activities will be performed in a process. It is represented with a solid line and arrowhead. |  |
| Message Flow | A Message Flow is used to show the flow of Messages between two Participants (between two pools). |  |
| Association | An Association is used to link information and Artifacts with BPMN graphical elements. |  |
| Pool | A Pool is a graphical representation of a Participant in a process. It contains one or more Lanes. |  |
| Lane | A Lane is a sub-partition within a Process, sometimes within a Pool that is used to organize and categorize Activities |  |
| Data Object | Data Objects provide information about what Activities require to be performed and what they produce. |  |
| Group | A Group is a grouping of graphical elements that are within the same category. |  |

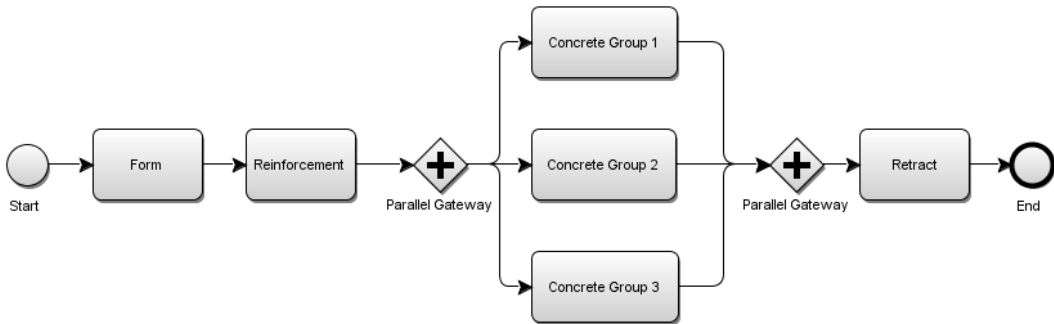


Figure 2-9: An example of BPMN Diagram

Comparison with other approaches

In the comparison with the UML, it is necessary to mention, that despite of the fact that both standards are maintained by the Object Management Group, they have a major difference. UML represents an object-oriented approach for a whole (software) system allowing describing it from different views. Only one type of its diagram (the activity diagram) can be suitable for the process modeling. BPMN to the contrary represents a process-oriented approach and was originally developed for the business process modeling.

One of the main differences with the EPC is that BPMN was strongly influenced by Workflow languages and has process automation as a primary focus (models are convertible to BPEL) and therefore is more suitable for the low-level process modeling, while the EPC mostly used for modeling higher level business processes.

2.3.5. Traditional modeling techniques

It is also necessary to mention here some traditional techniques, which can be used for the process modeling. These techniques are still widely used nowadays, however they were developed much earlier than techniques presented in the previous sections and did not have initially any IT-support. Some of these techniques are based on the graph theory and therefore a short introduction to graph theory is presented below.

Graph theory

Graphs can be used to represent many practical problems. They are used to model different types of processes in biological, information and physical systems.

A graph is a pair of sets $G = \{V, E\}$, where V is the set of vertices or nodes and E is the set of edges or lines (Wilson, 1979). If the set V consists of ordered pairs of vertices, the graph is called directed; otherwise the graph is called undirected.

Graph theory provides also a matrix representation of graphs (Wilson, 1979). The adjacency matrix of the graph $G = \{V, E\}$ is a $n \times n$ matrix $D = (d_{ij})$, where n is the number of vertices in G , $V = \{v_1, \dots, v_n\}$ and

$$d_{ij} = \text{number of edges between the two adjacent vertices } v_i \text{ and } v_j$$

In particular, $d_{ij} = 0$ if (v_i, v_j) is not an edge in G . The adjacency matrix of a directed graph G is $D = (d_{ij})$, where d_{ij} = number of arcs that come out of vertex v_i and go into vertex v_j .

An example of a directed graph with four vertices together with the adjacency matrix is shown in Figure 2-10.

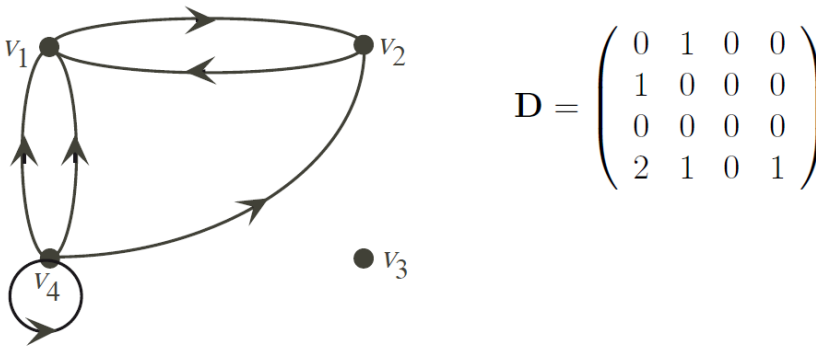


Figure 2-10: Graph and adjacency matrix

In civil engineering graphs can be used for different applications:

- Construction management (critical path analysis, network analysis);
- Process modeling (Petri nets);
- Planning of infrastructure (Shortest path problem);
- Force flow in a structural model.

With the help of graphs it is possible to model the basic structure of a specific system, as well as to check it for the consistency and completeness. In the following subsections Petri net modeling technique and critical path method, both based on graph theory, as well as some other techniques will be introduced shortly.

Petri nets

Petri net defined by Carl Adam Petri (Petri, 1962) in the beginning of the 1960s is a modeling technique including a mathematical formalization as well as a corresponding graphical representation. It is a basic model of parallel and distributed systems. A Petri net consists of places, transitions and arcs connecting them. It can be seen as a graph containing 2 types of nodes: circles (places) and bars (transitions). Places in a Petri Net represent possible states of the system and may hold tokens. An assignment of tokens to places is called marking. Transitions are events or actions which cause the change of state. A change of state is denoted by a movement of token(s) from place(s) to place(s).

An example of a simple Petri Net with three places (one with two tokens and other without tokens) and one transition is shown in Figure 2-11.

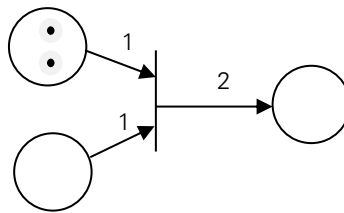


Figure 2-11: An example of a Petri net

Petri Nets as mathematical method possesses a number of analysis properties that can be used for the behavior analysis and for the verification of models. Such behavioral properties of models as reachability, boundedness and liveness (absence of deadlocks) can be checked with Petri Nets.

Critical path method

Critical path method (CPM) is developed in late 1950s and commonly used in project management. It provides a graphical view of the project and predicts the project duration. Additionally, it can show activities that are critical to maintain the project schedule and how long the complex project will take to complete.

Program Evaluation and Review Technique

Program Evaluation and Review Technique (PERT) is a project management technique, developed in the 1950s and mostly used together with the CPM. It is often used to schedule, organize and coordinate tasks within a project. PERT shows the total time required for the completion of the project as well as durations of each components (Archibald & Villoria, 1967).

For each modeled activity in PERT model different kinds of activity time are defined:

- *Optimistic time (OT)* – the minimum time in which an activity can be completed;
- *Most likely time (MT)* – the most probable time in which an activity can be completed;
- *Pessimistic time (PT)* – the maximum time which is required for an activity to be completed;

The *Expected time (ET)* for an activity can be calculated by using the following formula (2.1):

$$ET= (OT+4*MT+PT)/6 \qquad (2.1)$$

For the graphical representation a PERT chart can be used. A Pert chart example node is shown in Figure 2-12.

| | | |
|----------------|----------|-----------------|
| Early Start | Duration | Early Finish |
| Task Name | | |
| Late Start | Slack | Late Finish |

Figure 2-12: A PERT chart example node

PERT method is used to determine the critical path and to visualize critical dependencies between activities in a project, however it is necessary to note for the calculation that the actual distribution for the activity times can be different from the one assumed in the method.

Gantt chart

Gantt chart is a horizontal bar chart illustrating a project schedule. Gantt charts are widely used in project management, because they help to plan and coordinate tasks in a project. Nowadays they are supported by many software applications. Tasks in a project are represented by bars, which have corresponding to the task duration length. Gantt chart illustrates dependencies between tasks and can be used to critical and noncritical activities. An example of a simple Gantt chart is shown in Figure 2-13.

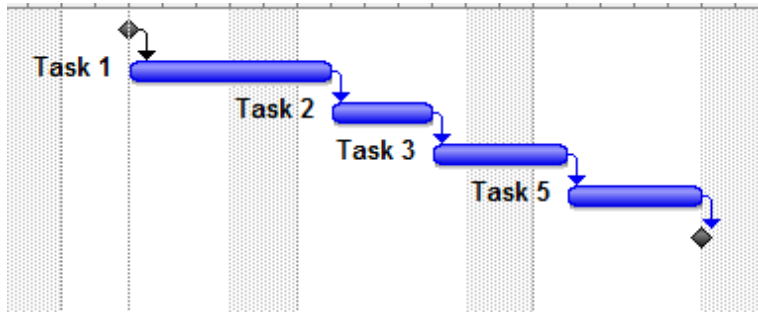


Figure 2-13: An example of a simple Gantt chart

2.4. Process modeling in construction industry

Process modeling in construction industry differentiates from the process modeling in general. Main features and applications of the process modeling in construction industry as well as some existing approaches will be considered in the following two sections.

2.4.1. Main features of process modeling in construction industry

Before discussing about the process modeling in construction industry it is necessary to define what construction process is. Halpin and Riggs (Halpin & Riggs, 1992) define a construction process as a unique collection of work tasks related to each other through a technologic structure and sequence.

Basically two main categories of processes can be distinguished in construction industry.

The first category includes processes that are not related to the construction process itself. These are processes governing the operation of an organization (management processes) and processes supporting the core processes in it (supporting processes). These processes usually structure and describe activities and tasks associated to management, accounting, recruitment and many other areas in the organization, that are not related directly to the manufacturing or production processes (core processes of the organization). These processes are being identified, formalized, modeled and analyzed in many companies in order to improve their efficiency and quality and, therefore, to increase competitiveness of the organization and to reduce costs. However, these processes are not specific for organizations from the construction industry only and they are common in a lot of other different areas. Therefore, in this section we focus on the second category of processes.

The second category includes processes associated with the core processes of an organization that are related directly to the construction process. These processes are quite different from many manufacturing or production processes in other branches. Despite some similarities construction industry can be only hardly compared with the manufacturing industry. In his work Jongeling (Jongeling, 2006) mentioned an example of a Swedish construction company NCC, that launched a residential building concept in spring 2006, where 90% of all components were preassembled in factories and then assembled on site, in a half time compared to traditional construction according to NCC (www.ncc.se). However, this case is rather an example of an exception that proves the rule, because most of the construction projects nowadays are still quite different from the manufacturing projects.

In civil engineering construction processes have a specific character. Each construction project is individual and is executed only once, which makes the main difference with manufacturing industry, where mass production is commonly applicable. Moreover, not only a large number of participating companies, but also different construction materials, machines, location and construction methods make construction very complicated to describe.

According to (Franz, 2010) following features, showing the uniqueness of the processes in construction industry, can be defined:

- Uniqueness of design and construction;
- Many problems and parameters with stochastic properties;
- Many different participants and stakeholders;
- Variety of mandatory and meaningful dependencies in the processes;
- Long periods of the process.

In Table 3 from (Franz, 2010) unique features of the construction processes in comparison with the stationary production are shown.

Table 3: A comparison of the construction processes with the stationary production

| Criterion | Stationary production | Construction site |
|------------------------|--|---|
| Manufacturing planning | Usually mass production, unique planning for many executions | Usually one-of-a-kind production, unique design for one-time execution |
| Production site | Production site at a manufacturing plant, protected | Changes for each new project and sometimes also during the project, strongly depends on the |

| | | |
|------------------------|---|---|
| | against weather conditions | weather conditions |
| Production | Fixed production flow, usually without disabilities | Usually disturbed production flow due to many unpredictable disabilities |
| Production time | Relatively short for series and contract, well-planned in advance | Not predictable, permanent interruption in the schedule possible, relatively long |

Construction process modeling can be used to plan, control and improve construction processes. Two main application areas of process modeling in construction are scheduling and simulation.

Scheduling

Construction project scheduling is widely used to control the entire construction project by determining the timing and sequence of the activities and allocation of resources to them over time. This can be done when different activities, with their durations and the required resources are identified and then formally represented. A number of well-known project management software such as MS Project⁵ or Primavera⁶ is used to create the construction schedule.

Two different methods, which are used to schedule construction work, are presented in (Jongeling, 2006):

- *Activity-based scheduling method* is applied in most of today's construction projects and is relied on the activity-based Critical Path Method. It is based on the identification and managing of a limited set of activities (the critical path) controlling the entire project. Today a lot of powerful and affordable software solutions support this scheduling method.
- *Location-based scheduling method* is well-suited for construction projects consisting of large amounts of on-site fabrication, involving continuous or repetitive work at different locations. It uses lines in diagrams to represent different types of work performed on specific location by a different construction crews. By analyzing different deviations on these diagrams, scheduling mistakes can be identified and minimized.

⁵ <http://www.microsoft.com/project> Retrieved 2012-02-10

⁶ <http://www.oracle.com/primavera/> Retrieved 2012-02-10

Simulation

Simulation is used to imitate the operation of a real-world process or system over the time. In the construction process simulation a discrete-event simulation model is usually used. It is defined as simulation model in which the state variables change only at those discrete points in time at which events occur (Banks, 2009). Construction process simulation is used for many different tasks, such as improvement of the construction processes productivity, evaluating and planning of the resource utilization and material usage, finding an optimal combination of resource and activities, optimization of the construction schedule and many others.

Moreover, construction process simulation is tightly coupled with the construction process scheduling and they are very often based on the same or very similar process models.

However, the complexity of the simulation systems and the enormous amount of time and cost for the developing of the simulations models for construction processes make their application in the construction industry not as common as of the construction project scheduling.

2.4.2. Existing approaches in construction industry

Relational algebra and graph theory

Despite the significant development in the area of the process modeling, the application of the process modeling techniques for construction management still remains the main topic for the research community rather than for the construction companies. This can be seen from the example of small construction companies, where nowadays only Critical Path Method and Gant Chart have found the use in everyday practice.

A process modeling approach using relational algebra and graph theory has been developed by (Huhnt W., 2005) (Huhnt & Enge, 2006) (Enge, 2010) (Enge & Huhnt, 2008). Huhnt and Enge model construction tasks as triples, consisting of an activity, a component and a state (s. Figure 2-14) (Huhnt, et al., 2010).

Components are parts of the building, like walls, floors, etc., on which activities are performed. States describe results of activities.

A component-type-based approach is used to model construction processes. The whole building is being decomposed into identifiable components that are associated with the corresponding activities.

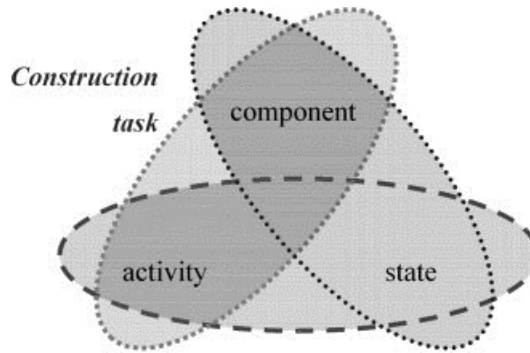


Figure 2-14: Construction task

A component type encapsulates the description of manufacturing process of a component and consists of a set of status values, predecessor-successor relations, unit efforts and stereotype placeholders for components. The user has to assign each component to a component type. As a result the components are then represented as independent graphs, describing the manufacturing process.

To model relations, connecting different graphs of various components, a so called activity-oriented approach is used. A construction process for the project “Garage”, consisting of five construction elements is shown in Figure 2-15 (Enge & Huhnt, 2008). An extended Event-driven process chain was used to model construction processes. Dashed lines between different processes are used to model prerequisites for the respective processes. For example the construction of the first wall (v_{18}) can started only after the first foundation is solid enough (e_4).

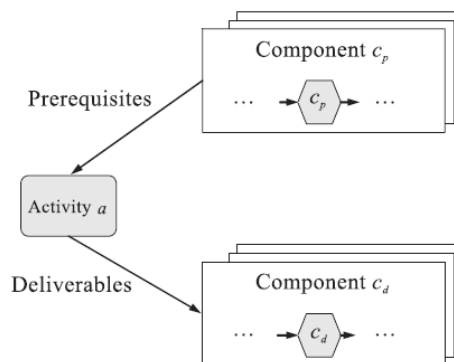


Figure 2-15: Activity-oriented process specification

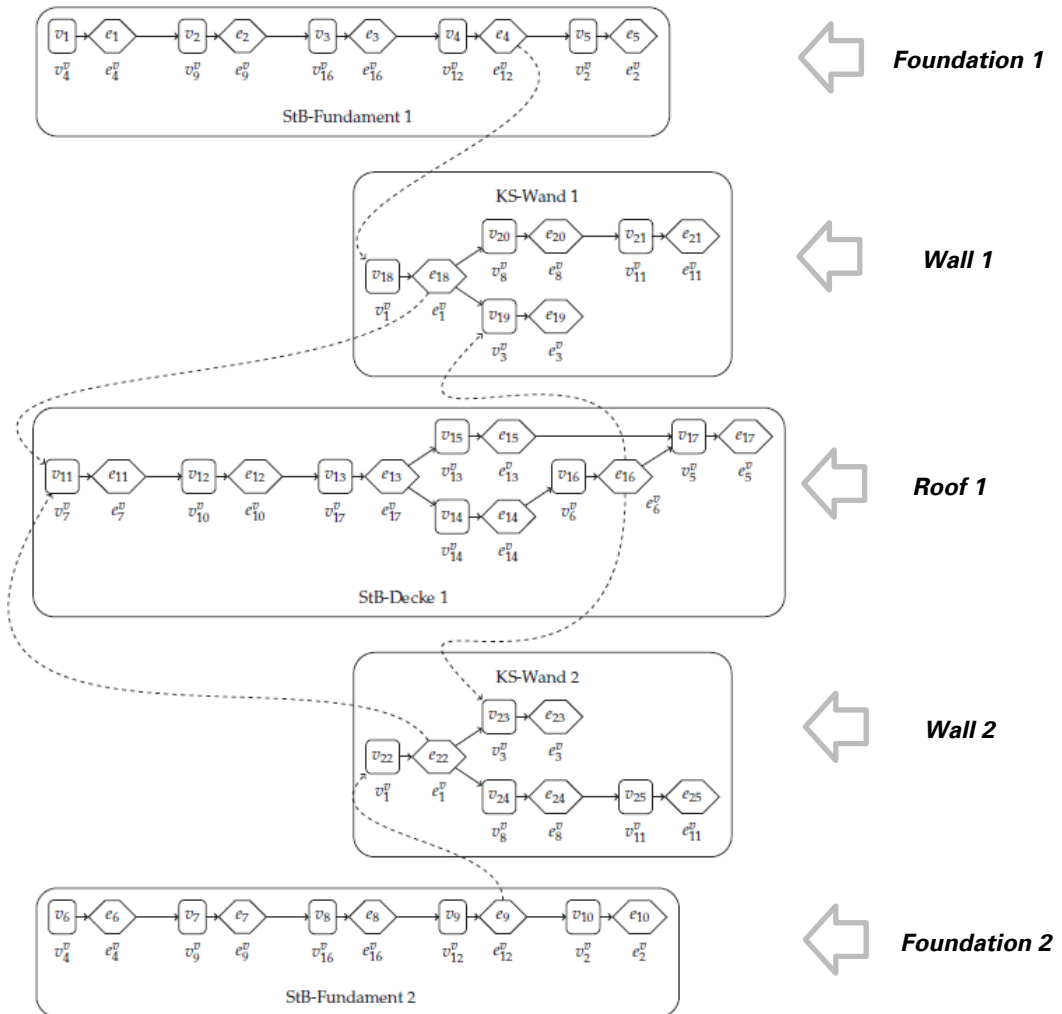


Figure 2-16: Construction process for the project "Garage"

As an example of this approach, a process, which models construction of a garage, consisting of five elements (two foundations, two walls and a roof), is shown in Figure 2-16 (Engel, 2010).

Modeling construction processes as graphs allows applying various mathematical algorithms for generation, optimization and validation of the construction process schedule.

One of the main benefits of the proposed approach is that the developed formalism based on set theory and relational algebra guarantees a consistent and correct structure of generated construction activities. Construction schedules can be generated semi-automatically, in such a way that the disassembling of the construction processes into

activities as well as the specification of the prerequisites are done by the user, but the technological interdependencies between activities are calculated by algorithm. (Huhnt W., 2005) (Huhnt & Enge, 2006) (Enge & Huhnt, 2008) (Huhnt, et al., 2010).

GEPM

A new GEneric construction Process Modeling method called GEPM was developed within the international MoPo (Models for the construction process⁷) research project in 1999-2002. This method was designed as a synthesis of the features of six existing process modeling methods: IDEF0, IDEF0v, IDEF3, scheduling, the simple flow method and Petri Nets. All these methods were developed for the specific purposes and therefore, their use for describing processes from viewpoints other than originally defined was quite difficult.

One of the main goals of this method was to overcome the deficiencies of the other methods and to support the transformation between various process model views. In Figure 2-17 from (Karhu, 2001) the concept of the generic construction process modeling method is shown. The principal concepts of every used method are written inside the circle. The largest circle covering all concepts represents GERM itself.

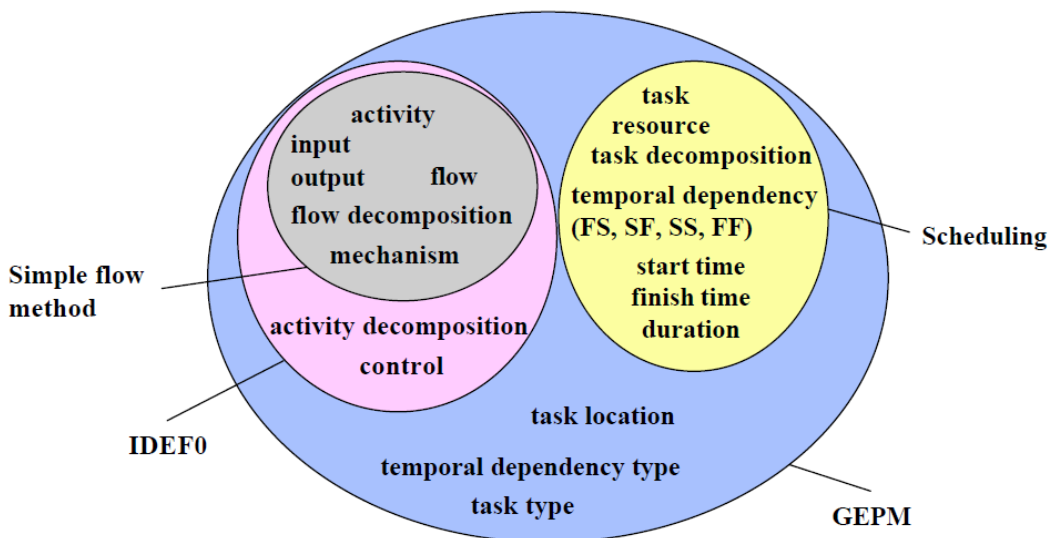


Figure 2-17: The concepts of the GEPM

⁷ http://cic.vtt.fi/vera/Projects/e_mopo.htm Retrieved 2013-08-25

The distinction between an activity and a task is one of the main purposes of GEPM. The definition of activity in GEPM is similar to the one in the IDEF0. Moreover tasks are defined similar to the tasks in the scheduling methods. Specific activities are called tasks. Both concepts are linked with the *task_type* relation. Flow objects can be used as input, output, control or mechanism of activities.

In Figure 2-18 from (Karhu, 2001) an extended conceptual model of GEPM as an EXPRESS-G Schema is shown. This model contains some additional concepts added to the first version from (Karhu, 2000).

A prototype application, called the GEPM browser has been implemented with the aim to test and evaluate the basic principles of the GEPM method. It has been used to create a complete GEPM model and to convert it to different views. The prototype was implemented on the Lotus Notes Platform⁸, where the most routines of the GEPM browser have been programmed with the object-oriented programming language Lotus Script.

The described Generic construction Process Modeling method provides an interesting example of the synthesis of a few existing methods for the modeling of construction process. It can be used in the construction process modeling to represent more features, then the other methods it is based on allow. The conceptual model can be changed in order to achieve additional features when needed. The usage of the MS Project for the scheduling and MS Visio for the drawing diagrams can guarantee the user-friendliness of the developed prototype, however at the same time makes the system dependent on these proprietary software tools and formats.

⁸ www.ibm.com/notes Retrieved 2013-08-28

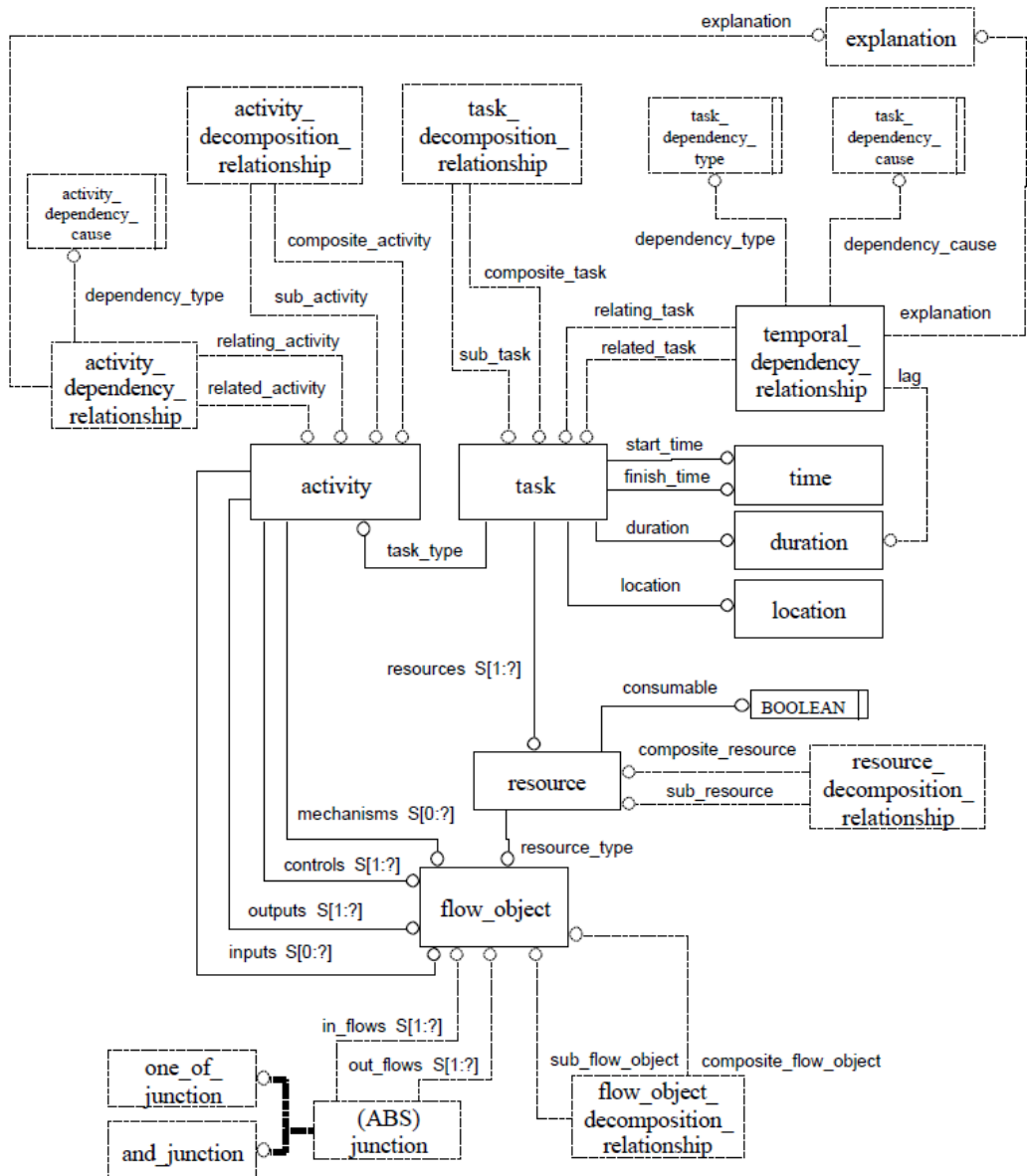


Figure 2-18: The conceptual model of the GEPM

Workflows graph and Petri Nets

The description of the construction processes by workflows graphs allows not only to describe these processes formally, but also to use various mathematical algorithms for checking the consistency and the correctness of the process model.

A Petri nets modeling technique, shortly introduced in the previous chapter are widely applicable for the process workflow modeling. A good overview of the application of Petri nets for workflow modeling is given in (Salimifard & Wright, 2001).

Within the research projects “Coordination of Planning Processes in Geotechnical Engineering” (Berkhahn, et al., 2005) and “Relational Process Modelling in Co-operative Building Planning” (Klinger, Berkhahn, & König, 2006) relational process models for planning processes in building engineering have been developed.

The basic idea of the proposed modeling approach is that the entire construction process is decomposed into activities and states. An activity describes a set of planning tasks and is handled by one or more participants. States describe relationships between activities. Both states and activities form a workflow graph that is formally described by a bipartite graph, where activities and states are nodes of this graph.

$$P := (A, S, R, Q, f_{AT})$$

$$\text{with } f_{AT}: (A \cup S) \rightarrow (A \cup S)_{AT}$$

A: Set of activities

S: Set of states

R: Relations between activities and states

Q: Relations between states and activities

f_{AT}: Mapping for the composition of activities and states

Because of the recursive decomposition of the initial process a hierarchical bipartite directed graph is used to represent the entire process structure. For the realization of parallel or alternative execution of activities few rules, similar to the standard logical operators XOR and AND, are introduced.

In that case some similarities with the event-driven Process Chain can be seen, where activities can be considered as functions and event as states. The main difference to EPC is that the hierarchical process structure is extended by a mapping for the marking on activities and states.

$$P := (A, S, R, Q, f_{AT})$$

with $m: (A \cup S) \rightarrow \{0, 1\}$

m : Marking of activities and states

This extension allows using the method of simple Petri nets for the hierarchical process structure. Activities and states can be marked with two values: 'completed' and 'not completed' for activities and 'active' and 'not active' for states. The consistency conditions for states are based on the firing rules of transitions in Petri nets. An example of the marked hierarchical process structure modeled as a bipartite directed graph from (Berkhahn, et al., 2005) is presented in Figure 2-19. It shows a consistent marking of a hierarchical process structure.

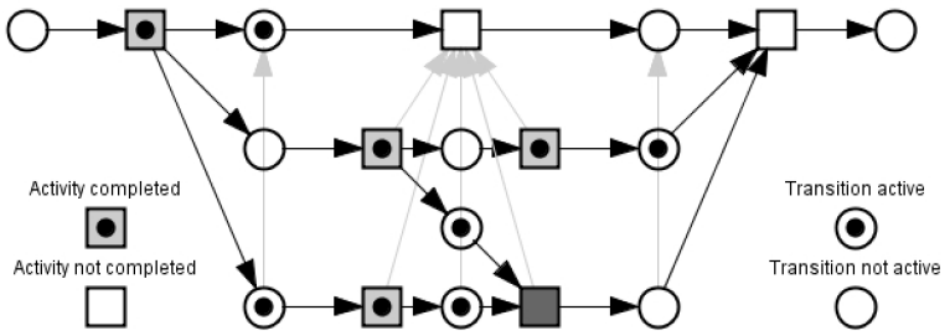


Figure 2-19: Hierarchical process structure

A software tool ProMise has been used to support process modeling and analysis based on Petri nets.

EPC

An event-driven Process Chain is used in many works to model construction processes.

In (Faschingbauer, 2011) this method has been used to model construction and monitoring processes in the domain of geotechnical engineering. In his work Faschingbauer combines process modeling and product modeling. This enables to link process model, represented with the core class 'activity' and a product model, represented with the core class 'object'. An example of such Product-Process Module from (Faschingbauer, 2011) for the production of a slurry wall is shown in Figure 2-20.

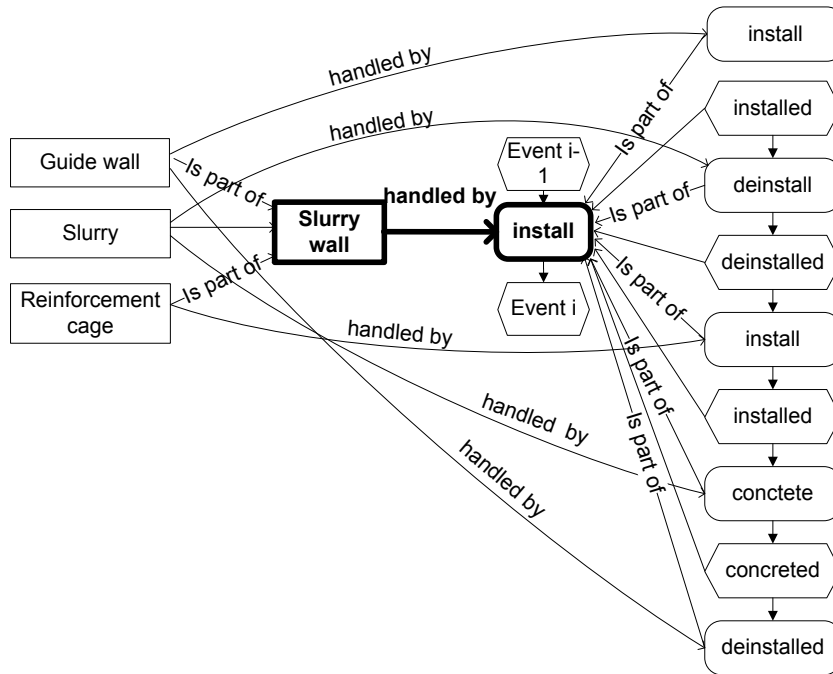


Figure 2-20: Product-Process Module for the production of a slurry wall

Within the research project Bauvogrid⁹ the Event-driven process chain has been used to model defect management processes in the construction projects. A pattern-based method using the ARIS methodology and extended Event-driven process (eEPC) chain was used as a baseline for the process modeling. Extended event-driven process chains are extended by some additional elements, that help to represent how the available resources implement a process as well as the way how the process interacts with the environment (Davis, 2001).

The problem of collaborative process management on the example of the Defect Management System (DMS) was investigated. The Business Process Object (BPO) concept (Scherer, et al., 2008) was introduced to enable the specification of the predefined reusable process patterns. A fragment of the eEPC for defect management is presented in Figure 2-21. It shows the responsibility of the General Contractor (GC) in the case when some defect is not a part of the contract (Scherer, et al., 2008).

Event-driven process chains are very suitable to represent standardized automated or semi-automated activities and have a good tool support with software products like ARIS or SAP NetWeaver, for example. However it is also necessary to note that, EPC is not a standard such as UML or BPMN are and is less popular outside of the German-

⁹ <http://bauvogrid.de> Retrieved 2013-08-26

speaking countries. Moreover EPC focuses only on process modelling while BPMN, for example, can be also used to create executable process models and therefore it can be seen as the bridge between business process modeling, implementation and execution.

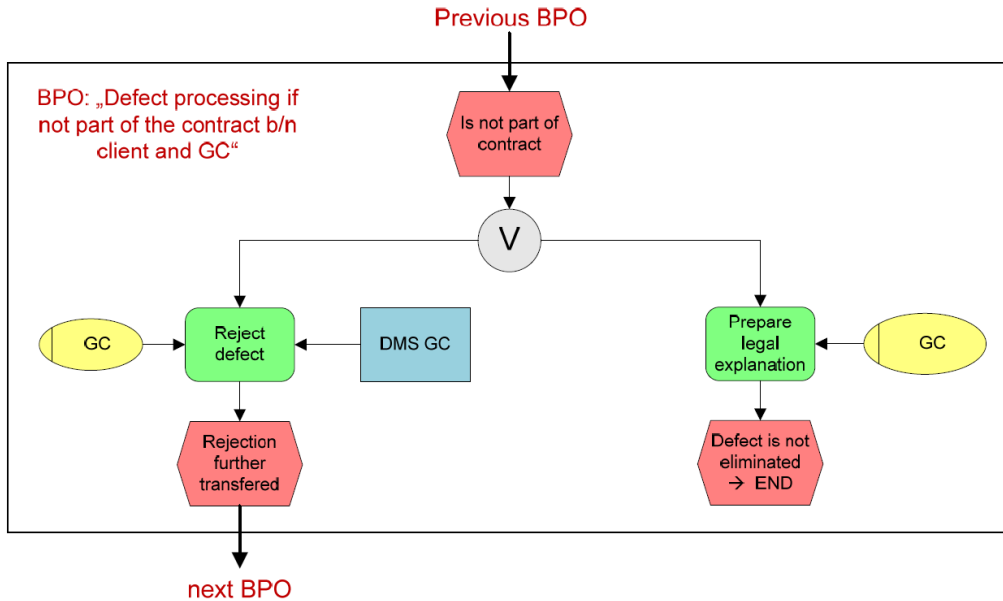


Figure 2-21: Example BPO from the eEPC for Defect Management

In (Sharmak, 2011) a configurable EPC, which is an extension of the classical EPC has been used to model configuration templates for exception handling in construction processes. The developed approach suggests the use of configurable fragments in the course of the process models in order to express the uncertain parts of the process. The configurable fragments are generalized and described on the type level as Configuration Templates. These templates are used as reference process fragments for the instantiation of risks responses in construction processes. In Figure 2-22 examples of the insertion templates are presented.

The template to the left shows, that the treatment is done before the risk evolves to a real problem, i.e. at least before the affected function Activity(n) starts. In the template shown to the right, Activity(n) is substituted with Activity(m) to eliminate the highly expected high risk impact. This alternative Activity(m), was not preferred in normal cases because of, e.g. its higher cost, its longer duration, or may be because it is technically more complicated to be executed (Scherer & Sharmak, 2008).

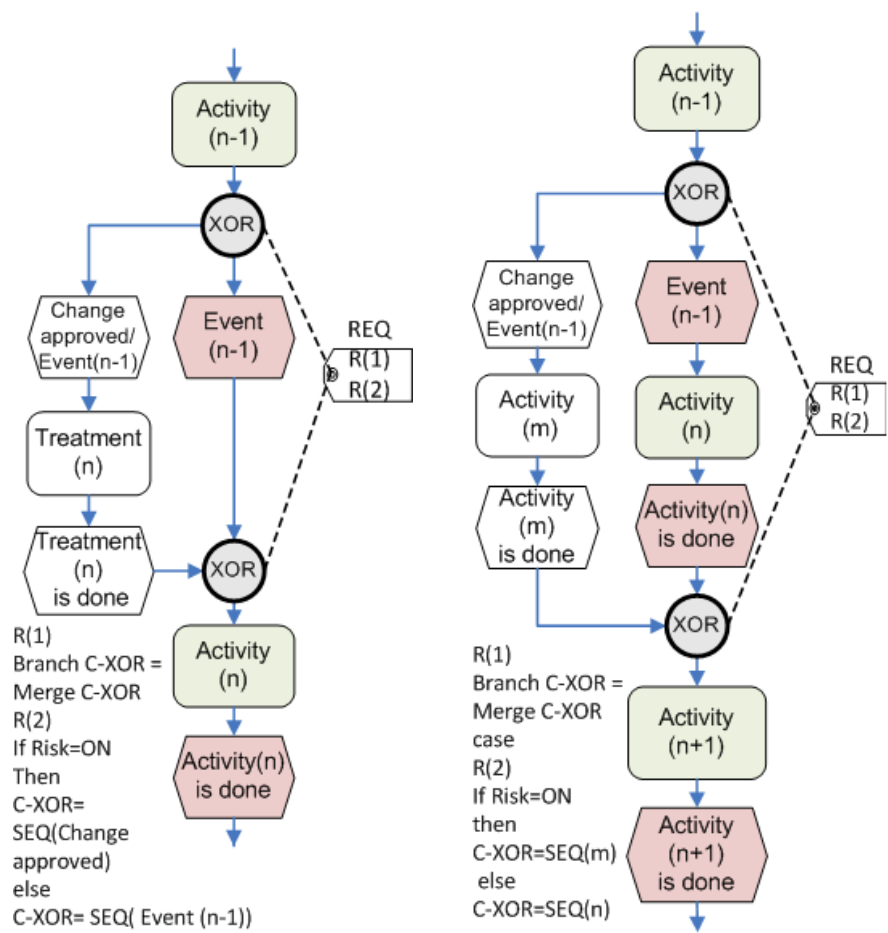


Figure 2-22: Risk treatment templates

IFCProcess

While discussing process modeling in construction it is necessary to mention the format proposed in the IFC specification. The IFC (Industry Foundation Classes) specification has been developed by buildingSMART and has its goal to represent an open specification to describe building and construction industry data. This is an open specification that supports various encoding of the data. Besides the default IFC exchange format that uses the STEP physical file structure according ISO10303-21, the format using the XML document structure and the format using the PKzip 2.04g compression algorithm are also supported¹⁰.

¹⁰ <http://www.iai-tech.org/products/ifc-overview> Retrieved 2013-10-25

The IFC Data model is primary used to describe a building model, including spatial structure, different building elements, their properties and relations between them. The core data schema map of the model¹¹, giving an overview of the modelled domain, is shown in Figure 2-23 .

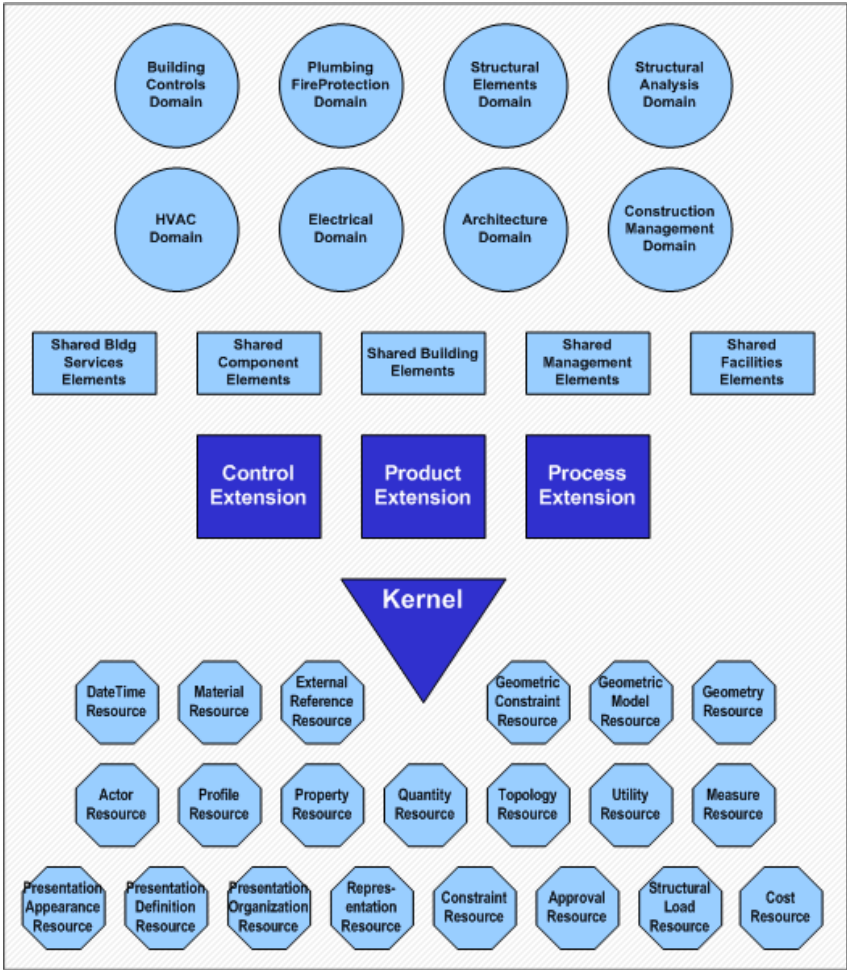


Figure 2-23: Core data schema map

Although the IFC data model is mainly used to describe the building model, it has some classes that support modeling of processes. Modeling of processes in the IFC model is similar to the product modeling and includes several objects and relations between them.

¹¹ <http://www.buildingsmart-tech.org/ifc/IFC2x4/rc3/html/index.htm> Retrieved 2013-10-25

In IFC the concept *IfcProcess* is defined as a set of activities that are interrelated or interact with one another. An *IfcProcess* can be an activity (*a task*) or an event. Resources are used in processes in order to transform inputs into outputs. Processes are interconnected in such a way, that outputs of one process are used as inputs for the following one.

Several relationships are defined in IFC to connect process to other objects. *IfcRelNests* allows building a hierarchy of processes by nesting them. With an *IfcRelSequence* relationship, processes can be placed in a sequence and have successors and predecessors. There are also some other relationships assigning processes to a work schedule (*IfcRelAssignsToControl*) or assigning products as input or output of the process (*IfcRelAssignsToProcess* and *IfcRelAssignsToProduct*). Resources such as labor, material and equipment can be linked to a process with an *IfcRelAssignsToProcess* relationship. The process relationships schema is shown in Figure 2-24.

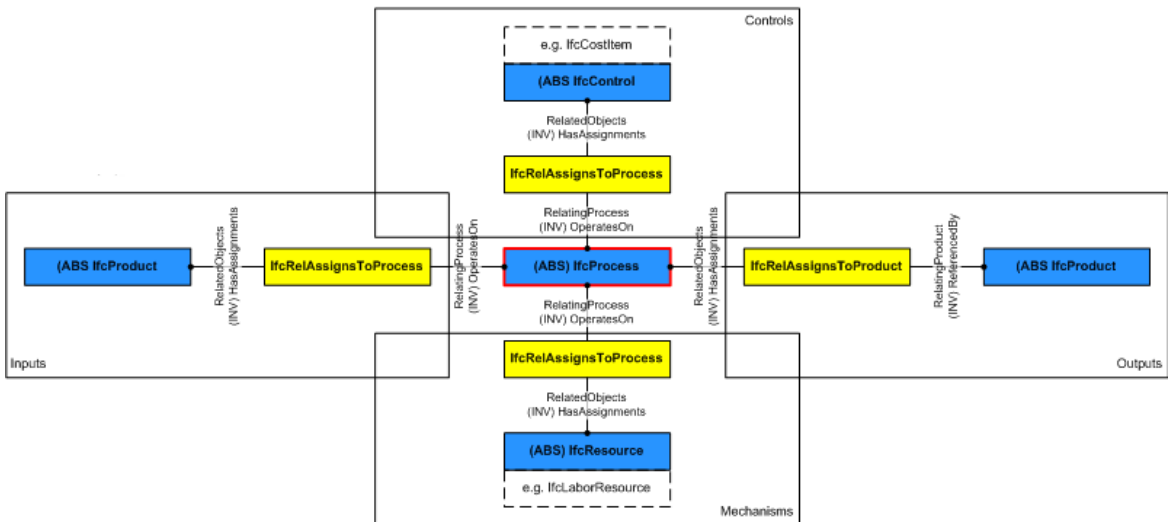


Figure 2-24: Process relationships diagram.

IfcProcess provides a prospective structure for modeling construction processes, however, despite the enormous support of IFC specification by many software vendors (such as Autodesk, Dassault Systemes, SolidWorks Corp, Nemetschek and many others), the main application aspect remains at the product modeling and not on the process modeling. With regard to the process modeling only few researches have been carried out and most of them were concentrating on the estimating of construction costs (Tanyer & Aouad, 2005) (Yabuki & Shitani, 2005) (Zhiliang, et al., 2011) and not on the modeling of processes.

Yabuki and Shitani developed a management system that facilitates cost estimating of earthworks and used IFC to represent the process model (Yabuki & Shitani, 2005).

An application and extension of the IFC standard in construction cost estimating was proposed in (Zhiliang, et al., 2011). In their work authors investigate the possibilities and methods of applying the IFC standard to the construction cost estimating for tendering in China. In order to describe the schedule of projects the entity *IfcProcess*, its subtypes and corresponding relational IFC entities were used. A planning tool for 4D model schedule simulation and cost estimation, that uses an IFC-based project database, is developed in (Tanyer & Aouad, 2005).

3. Ontology-based modeling

In this chapter the main concepts of the knowledge-based modeling and in particular the ontology-based modeling are presented. The first section discusses different approaches that can be used for the knowledge representation. It includes the concept of the Ontology Spectrum and its main components such as taxonomy, thesaurus, conceptual model, topic maps, logic theory and ontology.

A detailed description of the notion of ontology as well as a short overview and a comparison of the most common ontology description languages are given in the section two. The main focus in this overview is on the Web Ontology Language, as this language is used in this work to describe ontologies. Some main concepts of its logical base – the Description logics will be presented in a separate subsection.

The last section presents a literature survey about the use of ontologies for the representation of various processes.

3.1. Knowledge-based models

Before discussing what ontology is, it is necessary to consider other different approaches that can be used for the knowledge representation. The following subsection represents the concept of an Ontology Spectrum and its components such as taxonomy, thesaurus, topic maps and finally the ontology.

Taxonomy

Taxonomy provides a tree-like structure that is used to classify or categorize a set of things. In (Daconta, et al., 2003) the following definition of taxonomy is given:

“Taxonomy is a classification of information entities in the form of a hierarchy, according to the presumed relationships of the real-world entities that they represent.”

An example of a part of a simple taxonomy for processes is given in Figure 3-1.

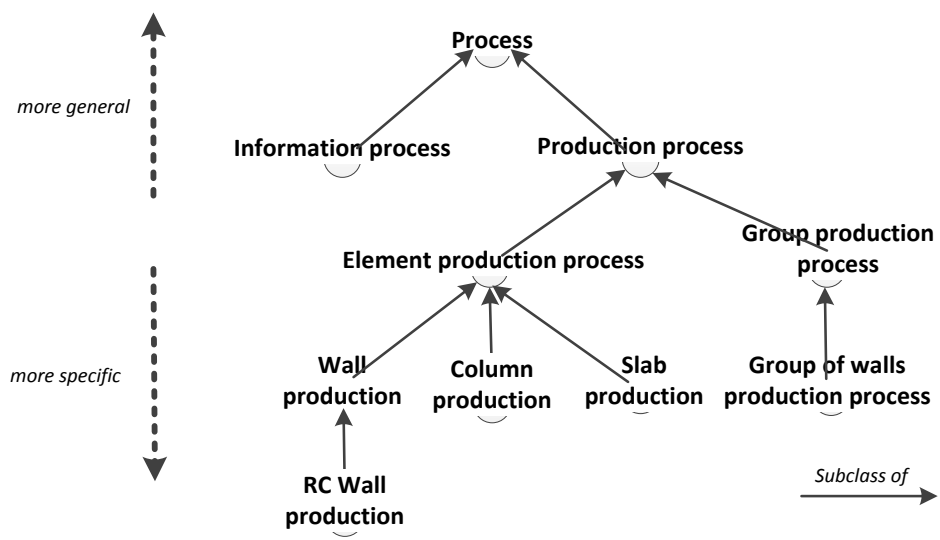


Figure 3-1: A simple taxonomy for processes

Having a tree form a taxonomy contains one root and several branches. Branching points are called nodes and represent information entities. Nodes are related with each other with a subclass relation. The higher we go in the figure the more general entities we become and less specific features we have (For example an entity *Process* is more general than an entity *RC Wall production*) and with the step down the entities become more specific.

Taxonomies have found their application in many areas. They play an important role in the biology where they are commonly used for the classification of species or plants with different degrees of generality. Nowadays they are also widely used in the area of information technologies for the classification of various products and services. Figure 3-2 shows a part of the taxonomy representing different production processes. This taxonomy is used as a basis for constructing the Process Pattern Ontology that is considered in the following sections.

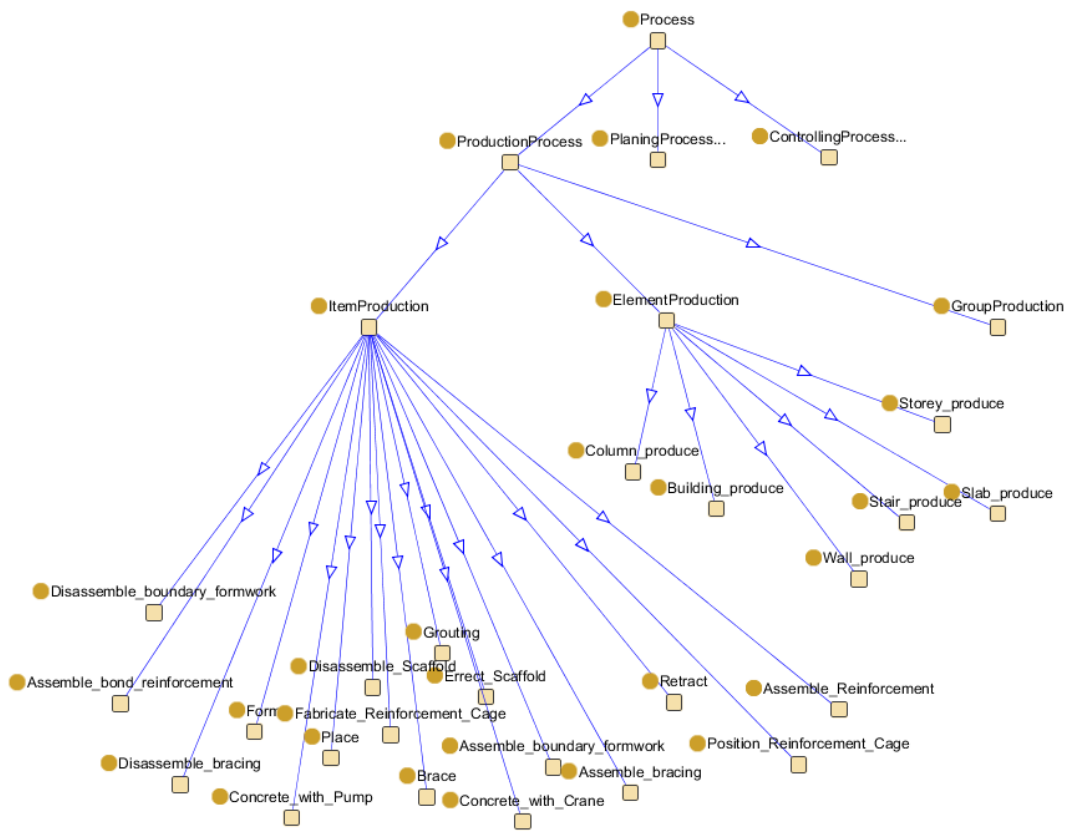


Figure 3-2: A taxonomy for production processes

Ontology spectrum

With the aim to compare the semantic richness of different knowledge-based models a so called "*Ontology Spectrum*" was introduced. The main idea of these knowledge-based models is to express concepts and relations between them in order to make the knowledge explicit. Ontology spectrum tries to represent these models in a general classification and displays the relations between them. There are different variants of the ontology spectrum (or sometimes also called "*semantic spectrum*") proposed by various author. In Figure 3-3 an ontology spectrum proposed in (Daconta, et al., 2003) is shown.

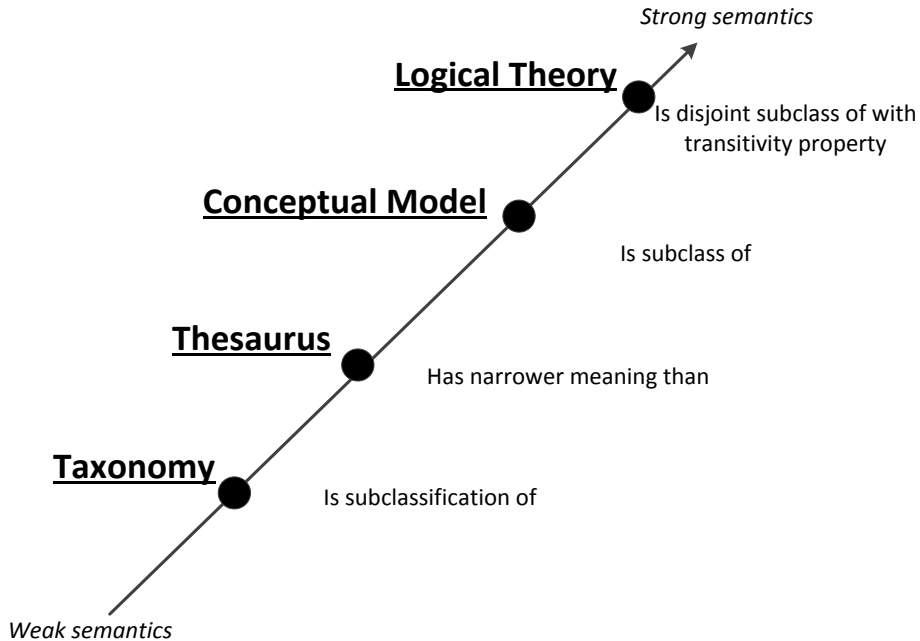


Figure 3-3: An ontology spectrum

By going from lower left to upper right part of the spectrum the semantic richness of the represented knowledge-based models increases and that is also the reason why the poles of the spectrum are called “*Weak semantics*” and “*Strong semantics*”. The first component of the spectrum – the taxonomy, was already described in the previous section. In the following sections other components of the ontology spectrum will be considered.

Thesaurus

A taxonomy is not able to express rich meaning of the concepts or provide some complex relations between them and therefore it is in general semantically weak. More semantic expressivity has a thesaurus that follows taxonomy in the ontology spectrum. A thesaurus is defined in (ANSI/NISO Z39.19-2005) as:

“A controlled vocabulary arranged in a known order and structured so that the various relationships among terms are displayed clearly and identified by standardized relationship indicators.”

In comparison to taxonomy, thesaurus provides some additional semantics for relations between entities. One of the most important extensions is a synonym relation that allows linking entities with similar meaning. In Table 4 from (Daconta, et al., 2003) a good overview of the semantic relations of a thesaurus is provided.

Table 4: Semantic relations of a Thesaurus

| Semantic relation | Definition | Example |
|--|---|--|
| Synonym Similar to Equivalent Used for | A term <i>X</i> has nearly the same meaning as a term <i>Y</i> | "Report" is a synonym for "document" |
| Homonym Spelled the same Homographic | A term <i>X</i> is spelled the same way as a term <i>Y</i> , which has a different meaning | The "tank", which is a military vehicle, is a homonym for the "tank", which is a receptacle for holding liquids. |
| Broader than (Hierarchic: parent of) | A term <i>X</i> is broader in meaning than a term <i>Y</i> | "Organization" has a broader meaning than "financial institution" |
| Narrower than (Hierarchic: child of) | A term <i>X</i> is narrower in meaning than a term <i>Y</i> | "Financial institution" has a narrower meaning than "organization" |
| Associated Associative Related | A term <i>X</i> is associated with a term <i>Y</i> , i.e., there is some unspecified relationship between the two | A "nail" is associated with a "hammer" |

Graphically a thesaurus can be represented similar to a taxonomy to which some additional relations were added. However it does not necessary have a tree structure.

The Wordnet¹² - a lexical database for the English language is one of the mostly known thesauruses in the area of information technology nowadays. It can be seen as a mixture of a normal dictionary and a thesaurus and is mostly used for many artificial intelligence and text analysis applications.

¹² <http://wordnet.princeton.edu> Retrieved 2012-11-25

Conceptual Model

A stronger semantics than a thesaurus provide Conceptual models. A Conceptual model represents entities as well as relationships between them and is common in modeling databases or applications. Different notations such as Unified Modeling Language (UML), Entity Relationship Model (ER model) or XML Topic Maps (XTM) can be used to describe conceptual models. A Class diagram is often used in UML to describe the Conceptual model. In the ER model the Conceptual model is usually described with an ER diagram. XTM provides XML syntax for describing Topic Maps that can be used to define Conceptual model. UML and ER diagrams are commonly used in the field of software engineering, while the main application of Topic Maps is representation and interchange of knowledge and therefore mainly this standard is used for describing conceptual models.

The XTM standard provides syntax for the interchange of Topic Maps and identifies their key concepts. Topic Maps appears to be similar to the thesaurus or taxonomy, however they have some additional features that make them semantically stronger.

The main concepts of Topic Maps are: topics, associations and occurrences.

- *Topics* are used to represent any kind of entities and are similar to nodes in taxonomy.
- *Associations* represent relations between topics and are similar to arcs in taxonomy graph.
- *Occurrences* represent information resources specifying relevant information for a particular topic.

In Figure 3-4 an example of a topic maps graph with its main concepts is shown.

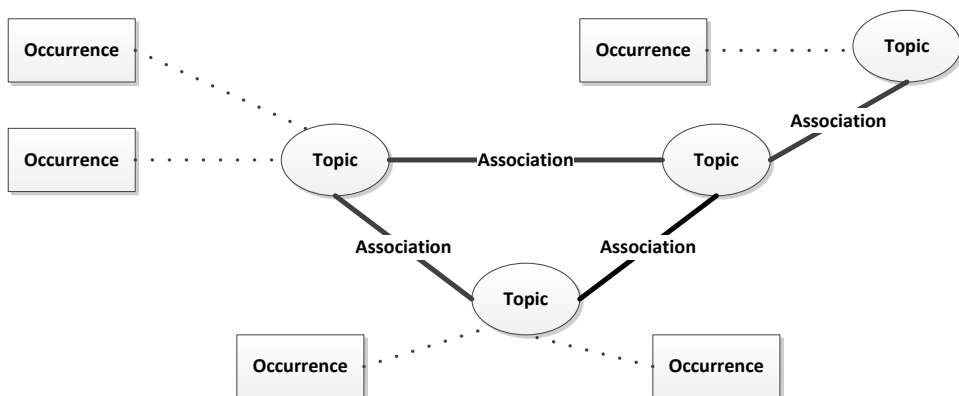


Figure 3-4: Topic map main concepts

It is necessary to note that semantic expressivity of Topic Maps is similar to that of the Resource Description Framework Schema (RDFS). Both models have the same central concept of an entity (a *Topic* in Topic Maps or a *Resource* in RDF/S), about which assertions are made. The main difference is that RDF/S relates one thing to another, while Topic Maps can relate any number of things and allow *n-ary* relations. A more detailed description of RDF/S will be presented in the following subchapter.

Logical theory

The most expressive component of the ontology spectrum can be found in its upper-right endpoint. It is called logical theory and it allows expressing the semantics of a model in the highest possible degree in comparison with other components of the spectrum.

Logical theories are built on axioms and inference rules that are used together to create new knowledge or prove theorems about the domain. Additionally to the *subclass of* relation defined in some other models of the spectrum, many new relations or properties (such as *disjoint of* relation or properties of symmetry or transitivity) are added. The main component of the logical theory - an ontology as well as its logical foundation and ontology description languages will be presented separately in the following section.

3.2. Ontology

In this section ontology, the semantically strongest component of the ontology spectrum will be defined.

Nowadays ontologies are widely used. Originally the term "*ontology*" comes from philosophy. Together with logic ontology is an important area of philosophy. "*Ontology*" is a big word in philosophy and it was used by many different philosophers in different ways, depending on their views.

In general ontology can be defined as a study of what there is. Many classical philosophical problems can be seen as ontological problems: the question of the meaning of being and the problem of the existence of universals are just some of them. The goal of the ontology in philosophy is to represent the world using its objects and connections between them.

Four parts of the large discipline of ontology have been defined in the Stanford Encyclopedia of Philosophy (Zalta, 2013):

- The study of ontological commitment, i.e. what we or others are committed to;
- The study of what there is;

- The study of the most general features of what there is, and how the things there are related to each other in the metaphysically most general ways;
- The study of meta-ontology, i.e. saying what task it is that the discipline of ontology should aim to accomplish, if any, how the questions it aims to answer should be understood, and with what methodology they can be answered.

The definition of ontology in computer science is quite similar. Here ontology serves as a formal representation of a set of concepts within a domain and the relationships between those concepts, where a domain is some area of knowledge or a subject matter area like construction, medicine, production, finance or many others. That means that ontology tries to capture the semantics of a particular domain and to characterize it through concepts and relations between them.

One of the reasons why computer scientists use ontology is that this representation can be processed by a computer. The real representation is usually very complex and consists of a lot of details that aren't necessary for the concrete goal, and therefore it has to be simplified and abstracted in such a way, that it can be understood by a machine.

Ontology typically describes following elements:

- Concepts (or classes) are general things in different domains;
- Individuals (or instances) are particular things;
- Relations among classes and individuals;
- Properties of the things (including their value);
- Rules or constraints involving those things.

Of course the concrete realization depends on the language in which ontology is expressed, but these elements are common for all realizations. In order to represent an ontology different knowledge representation languages, which allow clear and precise representation, are used. Natural language cannot be used for that purpose, because it is very ambiguous and therefore hard to process for the information technology use. Usually relations in ontology are represented through the hierarchy of classes, but they are not restricted just by them.

3.2.1. Ontology description languages

Ontology is encoded using ontology description languages. The most common languages that can be used for that purpose are: KIF (Knowledge Interchange Format), F-Logic, RDF(S) and OWL. All of them have different expressive power but have well-defined syntax, which makes them processable by computers.

Following requirements that can be applicable for the ontology description languages were summarized by Tim Berners-Lee¹³. These languages must:

- Have a compact syntax.
- Have a well-defined semantics.
- Have sufficient expressive power to represent human knowledge.
- Have an efficient, powerful and understandable reasoning mechanism.
- Be usable to build large knowledge bases.

In the following sections firstly some early languages for representing knowledge like the Knowledge Interchange Format and F-Logic are discussed briefly. Afterwards recent ontology description languages that have found their practical usage in the field of the semantic web and became a standard, such as the Resource Description Framework and Web Ontology Language (OWL) are presented.

Knowledge Interchange Format (KIF)

Knowledge Interchange Format (KIF) is a language designed to be used in the interchange of knowledge among disparate computer systems (Genesereth, et al., 1992). Its essential features are¹⁴:

- The language has declarative semantics. It is possible to understand the meaning of expressions in the language without appeal to an interpreter for manipulation those expressions.
- The language is logical comprehensive - it provides for the expression of arbitrary sentences in predicate calculus.
- It allows making all knowledge representation explicit and permit to introduce new knowledge representation constructs without changing the language.
- Translatability. It enables practical means of translating declarative knowledge base to and from typical knowledge representation languages.
- Readability. It has human readability facilities, although it is not intended primarily as a language for interaction with humans.
- Useability as a representation language.

KIF is based on first order logic and provides definitions for objects, relations, functions and logical constants. The Suggested Upper Merged Ontology (SUMO) is written in the SUO-KIF language, which was derived from KIF language to support the definition of this ontology.

¹³ The Semantic Web as a language of logic <http://www.w3.org/DesignIssues/Logic.html> Retrieved 2013-09-28

¹⁴ Logic Group Technical Report <http://logic.stanford.edu/kif/Hypertext/kif-manual.html> Retrieved 2013-09-22

An example of a KIF sentence “Every person has a mother” is shown below (3.1-5):

- (3.1) (*forall*(? *x*)
- (3.2) (\Rightarrow (person ? *x*)
- (3.3) (exists (? *y*)
- (3.4) (and (person ? *y*)
- (3.5) (mother ? *x* ? *y*))))

Frame Logic (F-Logic)

Frame Logic (F-Logic) is a logic language combining frame-based languages and first-order predicate calculus (Kifer & Lausen, 1989). In frame-based languages frames (or classes) are the central modeling primitives. F-Logic has a declarative and compact syntax and includes sound and complete proof theory. It allows representing classes, relations, individuals, functions, axioms and rules.

F-Logic was developed for the deductive databases, but is also widely used as formalism for ontologies. Nevertheless, nowadays description logic (DL), as well as, based on it, Web Ontology Language, (OWL) are used and accepted for the semantic web technologies more than F-Logic¹⁵.

An example of a F-Logic declaration is shown below (3.6-10):

- (3.6) driver :: person.
- (3.7) alex: person.
- (3.8) ferrari: car.
- (3.9) maserati: car.
- (3.10) drive[alex \rightarrow {ferrari, maserati}].

¹⁵ Reasoning on the Semantic Web, <http://reasoningweb.org/> Retrieved 2013-08-12

Resource Description Framework

The Resource Description Framework (RDF) is a standard model for the data interchange on the Web proposed by World Wide Web Consortium (W3C). It provides a standard form for representing metadata in the XML format. The main notion in the RDF description model is a triple or a statement. With triples it can provide a description model and syntax for representing different resources. Each triple consists of three elements interrelated with each other: Subject, Predicate and Object. The statement “Semantic Web is published by Springer” is represented in RDF in Figure 3-5.

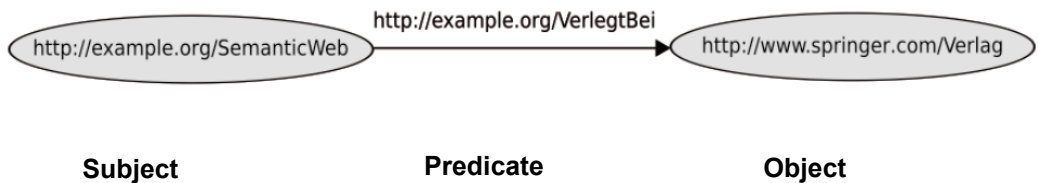


Figure 3-5: RDF triple

The RDF data model is an example of the classic conceptual model presented in the previous sections. The main modeling concepts of RDF are:

- Each resource has a unique identifier - URI (Universal Resource Identifier).
- Property (or predicate) can be seen as a special kind of resource describing relations between resources. They are also identified by URI.
- Statement (or triple) asserts the properties of resources.
- Each statement consists of a resource (subject), a property (predicate) and a value (object).
- Values can be resources or literals.

Nowadays there are different serialization formats for the RDF. The most widely used is an RDF/XML format, which also was defined as a main format for the RDF. In addition there are also two not-XML serializations that were designed to be easier written and understood by human such as Notation3¹⁶ and Turtle¹⁷.

¹⁶ <http://www.w3.org/TeamSubmission/n3/> Retrieved 2013-07-01

¹⁷ <http://www.w3.org/TeamSubmission/turtle/> Retrieved 2013-07-01

RDF is a universal language that allows describing resources. However, it is domain-independent and, therefore, cannot define semantics of any domain.

Resource Description Framework Schema

The Resource Description Framework Schema (RDFS) was built as an extension of RDF and provides a mechanism for describing a specific domain. While RDF can express only simple statements about resources using named properties and values, RDFS includes facilities for describing classes and properties and therefore it provides basic elements for describing ontologies.

The main concept in describing a particular domain is a class. RDF Schema allows creating class hierarchies and inheritance by using the "*SubclassOf*" relation between classes. Main concepts appearing in RDF Scheme are:

- Class;
- Subclass;
- Properties;
- Subproperties;
- Domain;
- Range;
- Restrictions.

Therefore RDF Schema can be seen as a primitive ontology language that has possibilities for describing domains and offers certain modeling primitives. In order to retrieve and manipulate data from RDF or RDFS a query language i.e. SPARQL can be used.

An example of a description of a simple process in the RDFS format is given in Figure 3-6. It shows a process with ID *Produce_id0G9sS81*, which describes the production of a column and is an instance of the class *Column_Produce*. This process has a reference to an object column with ID *Columnid0G9sS81* and has five subprocesses *Form*, *Reinforcement*, *Brace*, *Concrete* and *Retract* with IDs: *Form_id0G9sS81*, *Reinforcement_id0G9sS81*, *Brace_id0G9sS81*, *Concrete_id0G9sS81* and *Retract_Braceid0G9sS81*. Relations to the connected processes are modelled through *hasPrevious* and *hasNext* properties. From the example it also can be seen, that class *Column_Produce* is a subclass of the class *Element Produce*.

```
<rdf:Description rdf:about="http://www.mefisto-bau.de#Column_produce">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
<rdfs:subClassOf rdf:resource="http://www.mefisto-bau.de #Element Produce"/>
</rdf:Description>

<rdf:Description rdf:about="http://www.mefisto-bau.de#Produce_id0G9sS81">
<rdf:type rdf:resource="http://www.mefisto-bau.de#Column_produce"/>
<hasObject rdf:resource="http://www.mefisto-bau.de # Columnid0G9sS81"/>
<hasPrevious rdf:resource="http://www.mefisto-bau.de#Produce_id2PB9 "/>
<hasNext rdf:resource="http://www.mefisto-bau.de#Produce_id2iK3PA"/>
<hasSubTask rdf:resource="http://www.mefisto-bau.de#Form_id0G9sS81"/>
<hasSubTask rdf:resource="http://www.mefisto-bau.de#Reinforcement_id0G9sS81"/>
<hasSubTask rdf:resource="http://www.mefisto-bau.de#Braceid0G9sS81"/>
<hasSubTask rdf:resource="http://www.mefisto-bau.de#Concrete_id0G9sS81"/>
<hasSubTask rdf:resource="http://www.mefisto-bau.de#Retract_id0G9sS81"/>
</rdf:Description>
```

Figure 3-6: A simple process in the RDFS Format

Nevertheless, RDF and RDFS are still quite limited and many desirable modeling primitives are missing. A richer ontology language OWL will be considered in the next section.

Web ontology language (OWL)

The expressive power of RDF and RDF Schema is still quite limited because they allow the representation only of some ontological knowledge. With these languages it is possible to express subclasses and properties hierarchies with their domain and range definitions. However, there is no possibility to express some specific properties of classes like disjointness, symmetry, Boolean combination or cardinality restrictions.

In this work a Web Ontology Language is used to describe ontologies. It is a standard from the W3C (McGuinness & van Harmelen, 2004) and is nowadays the most widespread ontology language. OWL extends RDF/S expressivity by providing additional vocabulary for describing properties and classes. It includes some logic primitives like universal or existential quantifiers as well as possibilities to make restrictions on properties.

OWL and RDF look very similar, but OWL is a stronger language with greater machine interpretability than RDF/S. OWL is built on top of RDF/S, but it comes with a larger vocabulary and stronger syntax than RDF/S.

OWL has three sublanguages with different level of expressiveness and each of them is specified to fulfill a various set of requirements (Figure 3-7).

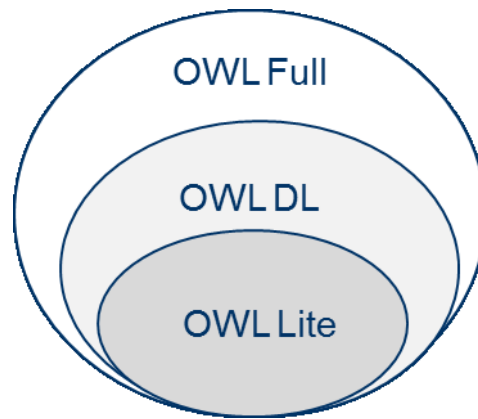


Figure 3-7: OWL sublanguages

OWL Full - is the entire language and it uses all the OWL primitives. It is fully upward-compatible with RDF, therefore every legal RDF document is also an OWL Full document. OWL Full was designed to preserve compatibility with RDF Schema. However, the language is so powerful, that it is undecidable and, hence, there is no reasoning software, that can be able to support every feature of OWL Full.

OWL DL - (stands for OWL Description Logic) is a sublanguage of OWL Full. It should be used when the maximum expressiveness without losing computational completeness and decidability is needed. The disadvantage is that the full compatibility with RDF is lost.

OWL Lite - restricts OWL DL to a subset of the language constructors. It supports a classification hierarchy and simple constraint features, but excludes arbitrary cardinality, disjointness statements and enumerated classes. The advantage of OWL Lite is that it is easy to implement, but we have to pay for this by its limited expressivity. It is the simplest language from the OWL family, but it is still more expressive than RDF/S.

These three sublanguages are upward compatible (McGuinness & van Harmelen, 2004):

- Each legal OWL Lite ontology is a legal OWL DL ontology.
- Each legal OWL DL ontology is a legal OWL Full ontology.
- Each valid OWL Lite conclusion is a valid OWL DL conclusion.

- Each valid OWL DL conclusion is a valid OWL Full conclusion.

With the specification from 2009 an extension and revision of the OWL, called OWL2 has appeared. OWL2 extends OWL and is backward compatible with it. It introduces 3 sub-languages, designed for different use cases and includes some new features that can be useful for some specific cases (like disjoint union or negative assertions).

In Figure 3-8 the relations between different ontology description languages, logics, semantic web rule language and the first order logic are shown. The expressiveness of languages increases by going bottom-up.

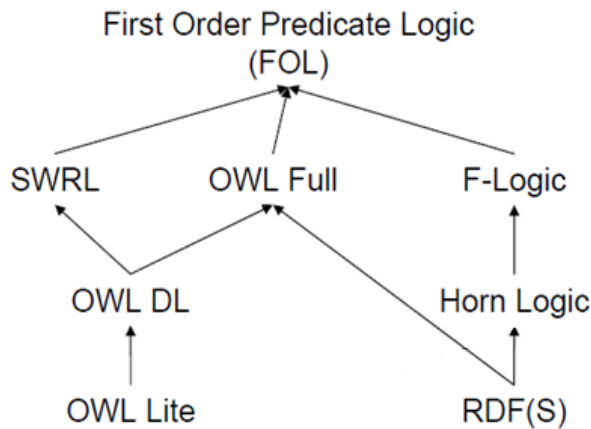


Figure 3-8: Relationships between different ontology description languages

Semantic query language

In order to retrieve and manipulate data stored in an ontology formalized in OWL a specific semantic query language is necessary. SPARQL (recursive acronym for SPARQL Protocol and RDF Query Language) is an official W3C Recommendation (since 2008) and plays for the ontologies the same important role as SQL (Structured Query Language) plays for the relational databases.

SPARQL query contains a set of triple patterns called a basic graph pattern. These patterns are similar to the RDF triples with an exception that some elements of the triple (subject, predicate or object) may be variables.

An example of simple SPARQL query consisting of a triple pattern with two variables (*?process* and *?subprocess*) is shown in Figure 3-9, where a search for production processes (search through metadata value) and their subprocesses is performed.


```
PREFIX test: <http://benevolenskiy.com/testOntology#>
SELECT ?process ?subprocess
WHERE {
    ?process :hasMetadata :Production.
    ?process :hasSubTask ?subprocess .
}
```

Figure 3-9: A simple SPARQL query

Main elements of a SPARQL query are:

- PREFIX that is used to avoid name conflicts by providing uniquely named elements and attribute in XML format, which is used for the serialization of RDF/OWL.
- SELECT is a clause, where variables appearing in the query results are identified.
- WHERE is a clause providing the basic graph pattern to match against the data graph.

Description Logics

All ontologies in this work are formalized in OWL DL, the Web Ontology Language based on Description Logic. DLs are a family of knowledge representation formalisms that can be used to represent and reason about concepts and relations between such concepts in a formally understood way (Baader, et al., 2003).

Elementary descriptions in DL are atomic concepts and atomic roles. Complex descriptions can be built from them inductively with concept and role constructors. A common DL level of expressiveness as basically available in OWL is *ALCQI*.

ALC (Attribute Language with concept negation) stands for a DL that allows only negation, conjunction, disjunction, and universal and existential restrictions, *Q* stands for number restrictions, and *I* for inverse roles. An abbreviation *S* is often used for the *ALC* with transitive roles. Therefore it is often said that OWL DL is based on *SHOIN(D)*, which is *ALC* with transitive roles together with role hierarchy (*H*), nominals (*O*), inverse properties (*I*) and cardinality restrictions (*N*).

The main constructs available in *ALCQI* are listed in Table 5 below.

Table 5: Syntax and semantics of ALCQI description logic

| Name | Syntax | Semantics |
|-------------------------|--------------------|--|
| Top concept | \top | Δ^I |
| Existential restriction | $\exists r.C$ | $\{x \in \Delta^I \mid \exists y.(x, y) \in r^I \wedge y \in C^I\}$ |
| Universal restriction | $\forall r.C$ | $\{x \in \Delta^I \mid \forall y.(x, y) \in r^I \rightarrow y \in C^I\}$ |
| Negation | $\neg C$ | $\Delta^I \setminus C^I$ |
| Conjunction | $C \sqcap D$ | $C^I \cap D^I$ |
| Disjunction | $C \sqcup D$ | $C^I \cup D^I$ |
| At-least restriction | $(\geq n \ r \ C)$ | $\{x \in \Delta^I \mid \#\{y \in C^I \mid (x, y) \in r^I\} \geq n\}$ |
| At-most restriction | $(\leq n \ r \ C)$ | $\{x \in \Delta^I \mid \#\{y \in C^I \mid (x, y) \in r^I\} \leq n\}$ |
| Inverse role | r^- | $(r^I)^{-1}$ |

The semantics of *ALCQI* concepts is defined in terms of an interpretation. An interpretation I consists of a non-empty set Δ^I (the domain of the interpretation) and an interpretation function, which assigns to every atomic concept A a set $A^I \subseteq \Delta^I$ and to every atomic role R a binary relation $R \subseteq \Delta^I \times \Delta^I$.

A knowledge base (KB) in DLs consists of two components: Assertion Box (ABox) and Terminological Box (TBox) (see Figure 3-10).

The ABox contains assertions about the named individuals, while the TBox introduces the vocabulary of an application domain and contains universal statement.

A general concept inclusion \sqsubseteq is used to specify subsumptions. For example the following sentence expresses that every construction worker is a person (3.11):

$$\textit{Construction_Worker} \sqsubseteq \textit{Person} \quad (3.11)$$

The same sentence can be written equivalently in the syntax of the First Order Logic as (3.12):

$$\forall(x)(\textit{Construction_Worker}(x) \rightarrow \textit{Person}(x)) \quad (3.12)$$

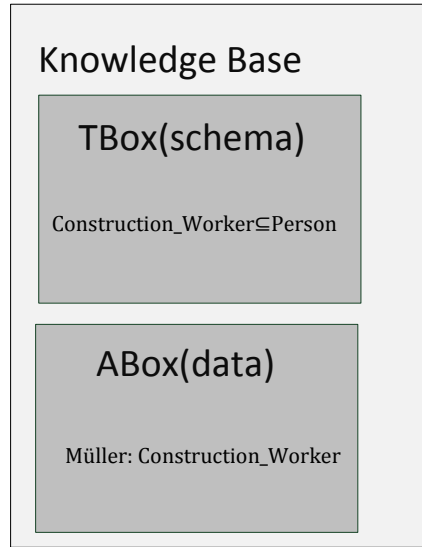


Figure 3-10: Knowledge base

Ontologies for processes

With regard to the ontological representation of business process models there exist multiple works such as (Osterwalder, 2004) (Gailly & Poels, 2007).

Ontologies have been employed for modeling processes and products in different fields including construction management, process scheduling, project design, cost estimation, risks management, etc. A policy-based framework for business process compliance was proposed in (El Kharbili & Pulvermüller, 2009). It uses several ontologies for the modeling purposes and demonstrates the advantages of semantic technologies for compliance management.

In construction, the need for the formal representation of the domain grew in recent years. In (Turk Z., 2006) the requirements were formulated for using ontology for construction industry. In the construction industry the concept of Process is presented as part of the domain-specific model IFC (Industry Foundation Classes), which is also available in the ontology form. The *ifcProcessExtension* includes the concepts *ifcProcess*, *ifcTask* etc. and attributes that demonstrate the essence of development processes.

In the frames of e-COGNOS project an ontology classifying construction concepts such as: process, product, project, actor and resource was proposed. It was developed in the process-centric manner and was used to support several application of knowledge management of the e-COGNOS project (El-Diraby, et al., 2005).

Similar main concepts for the construction ontology (process, actor, project, resource, method and failure) were used to describe production processes in (Moonseo, et al., 2013). These concepts comprise the top level of the construction ontology, which was used as a basis for the knowledge retrieval system.

An example of construction ontology that is Industry Foundation Classes compliant was demonstrated in the Semantic Web-based Information Management System (SWIMS) project (Pan, 2006). Wang (Wang, et al., 2010) used the ontology for the presentation of context-sensitive building information and the reasoning about it. This is the alternative way for construction information management. In the project InteliGrid (Katranuschkov, et al., 2006) (Dolenc, et al., 2007) the first attempts to formalize processes in Business Process Object Ontology were undertaken.

As part of the project eConstruct (van Rees, 2006), Building Construction Ontology Web (bcoWeb) was proposed, which should provide a common vocabulary for the construction industry. Additionally, different process modelling languages (mainly semi-formal) representing the ontological form, should be taken into account. Business Process Model (BPM) and Notation (BPMN) are formalized using OWL in (SUPER Project, 2010) (Ghidini, et al., 2008). As a basis for the formalization an OMG specification is taken.

A Petri net ontology for semantic description of concepts and relations has been proposed and modeled using OWL (Koschmider & Ried, 2005) (Gasevic & Devedzic, 2007). Semantic representation of the Event-driven Process Chain (EPC) has been proposed in (SUPER Project, 2010) (Kindler, 2006) and formalization of UML activity diagrams into OWL was shown in (Noguera, et al., 2010). Finally, the language-independent process (both EPC and BPMN) was formalized using OWL (Thomas & Fellmann, 2009).

The role of ontologies in the construction industry, where they support different aspects of design, production or scheduling, became more important in the last years. The mentioned works represent only some approaches. A good review of over 120 articles from the last ten years on built environment Semantic Web applications is presented in (Abanda, et al., 2013). The study investigates the actual trends and demonstrates the tendency of using more innovative Semantic Web technologies in the last years. While in the early years relatively simple ontologies were developed, the later years show the using of advanced Semantic Web technologies in the development as well as merging them e.g. in the construction industry domain with other technologies like BIM (Jernigan, 2007) or Linked Data¹⁸.

¹⁸ <http://linkeddata.org/>

In the following chapters main concepts and the structure of the two in this research developed ontologies are presented. Even that some ontological concepts like process, resource or product are common for many existing approaches, the ontological structure as well as the overall methodology are quite specific and novel.

4. Modeling of construction processes and process patterns

This chapter begins with the introduction of the reference modeling approach for the process modeling. Characteristics of reference models are presented with the benefits of their use.

The second section is dedicated to the description of the ontology-based process patterns. It includes the general requirements on the process patterns as well as the description of the principal structure of the patterns. Some examples of the process patterns are also given here.

The description of the ontological framework is given in the last section. It includes the discussion about the choice of an appropriate data model for the pattern representation and a comparison of ontology and database.

Two ontologies with their main concepts are considered in the last subsections. The first one is the Process Pattern Ontology that is used to store reusable process patterns. It can be seen as the TBox in the knowledge-based approach. The second is the Process Instance Ontology, which has similar taxonomy to the Process Pattern Ontology, but is uniquely populated with specific process assertions for each construction case and can be seen as the ABox in the knowledge-based approach

4.1. Reference modeling

Reference modeling is a promising approach for the process modeling that can speed up the design and implementation process by the reutilization of the knowledge contained in the reference models. This approach is based on reference models (sometimes also called generic models, universal models or model patterns) that are usually focused on a specific application domain and that can be used as reusable solutions for the development of new models.

There are many definitions of the reference modeling in the literature. Fettke and Loos defined in (Fettke, et al., 2003) a reference model, as a model that can be useful for the development of an individual model of an organization in some specific domain.

In the work of Vom Brocke (Vom Brocke, 2003) a reference model is defined as an information model that people develop or use for supporting the construction of applica-

tion models, where the relationship between the references and application models can be characterized by the fact that object or content of the reference model is reused during construction of the object or content of the application model (original German text: "Ein Referenzmodell ist ein Informationsmodell, das Menschen zur Unterstützung der Konstruktion von Anwendungsmodellen entwickeln oder nutzen, wobei die Beziehung zwischen Referenz- und Anwendungsmodell dadurch gekennzeichnet ist, dass Gegenstand oder Inhalt des Referenzmodells bei der Konstruktion des Gegenstands oder Inhalts des Anwendungsmodells wieder verwendet werden").

In (Rosemann, 2003) following definition of the term reference model is given: "Reference models are generic conceptual models that formalize recommended practices for a certain domain".

In their work (Castano, et al., 1998) authors write that "Reference components provide normalized descriptions of key concepts of a given domain. They can be used as the starting point for developing a new application similar to applications developed before in the domain, thus extending reuse to the early development phases. Reference components can also be used as a validation tool, to check the quality of existing schemas and promote their standardization."

As it can be seen from these definitions there are some characteristics of reference models that are common for all definitions:

- Reference models are universal. They represent a class of domain and therefore can be universally applicable;
- Reference models provide best practices;
- Reference models are reusable. They can be reused in multiple implementations.

The application of reference modeling has many benefits and can improve the modeling process because:

- Reference modeling can accelerate the development process and decrease the modeling time. Using reference model as a basis for modeling is faster than developing a new model from a scratch.
- Reference models can reduce costs, because development as well as production costs can be saved due to repetition and learning curves.
- Reference models can reduce risks, because they are already validated.
- Reference models improve the quality of models, because they represent normally proven solutions from the best practices.

In (Fettke & Loos, 2005) Fettke und Loos consider reference modeling as a combination of two main processes: construction process of reference models and application process of reference models. These two processes with their stages are shown in Figure 4-1.

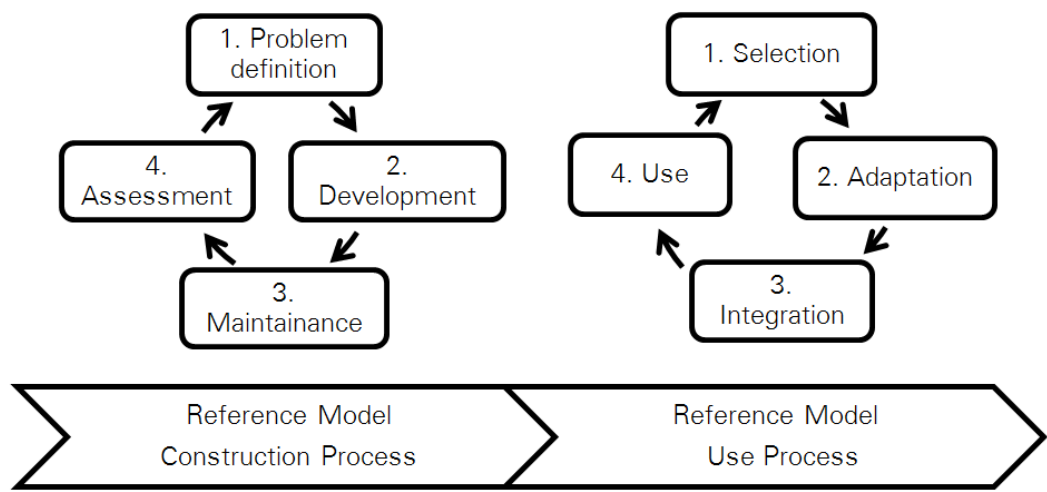


Figure 4-1: Reference Model Process

The first process consists of four stages (Problem definition, Development, Maintenance and Assessment) and has its goal to design and build a particular reference model. The second process, consisting also of four stages (Selection, Adaptation, Integration and Use) shows the reuse of the reference model for the development of a particular model.

In the following sections ontology-based process patterns will be described. These patterns can be seen as reference models from the domain “construction structural work” representing some typical construction processes. These patterns are used for the modeling and the configuration of some construction processes, however they are not restricted to the construction domain and can also be adopted with some extensions for other application areas.

At first the general structure of the pattern as well as a knowledge base for them will be introduced (that part corresponds to the first step of the Reference Model Process from Figure 4-1- Reference Model Construction Process). Then the use of the process pattern, which includes the process pattern retrieval and configuration will be shown (this part corresponds more to the second step of the Reference Model Process – Reference Model Use Process).

4.2. Process pattern

In this work the term *process pattern* will be used to describe reference models implemented for the domain “construction structural works”. The proposed process patterns comprise all main characteristics of the reference models:

- Process patterns are universal, because they describe not a particular instance, but a domain class.
- Process patterns defined in this work provide best practices, because they describe typical standardized construction processes.
- Process patterns are reusable, because they can be instantiated and reused for many different construction projects.

The first task in the design of the process pattern is to define their goal. For that it is necessary to determine how process patterns are going to be used and which requirements they must satisfy. Following requirements on the process patterns were formulated:

- Process patterns should provide a general structure for describing typical construction processes.
- The structure of the process pattern should be extendable, so that it can be also used to describe processes from other domain if necessary.
- Process patterns should be stored in the reliable knowledge base providing a powerful search mechanism for the pattern retrieval.
- Process pattern can be imported or exported in one of the standardized interchange format.
- Process patterns can be easily instantiated, adapted and used for the configuration of processes.

Process patterns proposed in this work basically include information of about three groups of elements. These include: required resources, related subprocesses and an applicable object. Because of the fact, that the application field of process patterns from this work is the domain of construction process, only building objects will be considered as applicable objects, however the overall structure and methodology also can be used to describe any other types of objects. The principal structure of the process pattern is shown in Figure 4-2.

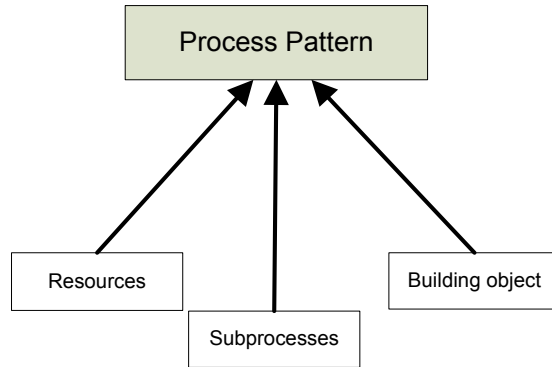


Figure 4-2: The principal structure of a process pattern

The proposed process pattern can be seen as task- or activity-oriented process pattern, because it describes detailed steps (subprocesses) to perform a specific task and includes all required resources. More detailed structure of the process pattern is shown in Figure 4-3.

Process pattern consists of following elements:

- *Description* contains a general description about the process pattern and its usage.
- *Meta-information* contains some metadata related to the process patterns, which can be the information about when the process pattern was created, who is the author, short description of the pattern or some tags that can be used for the quick search for a pattern.
- *Object* represents a building object, for which a pattern was created. These are usually some single building elements like a wall or a column, but that could be also a group of elements.
- *Parameters* describe different parameters related to the process pattern. There are two main groups of parameters that are considered: costs related parameters and time related parameters. These parameters are not used in the configuration process itself in the implemented approach. A proposal how to use these parameters in the extended configuration process is described in the outlook section of chapter 8.
- *Subprocesses* represent all processes that need to be done in order to perform the task described in the process pattern. Not only processes but also their logical order and some of their parameters are saved in the process pattern.
- *Resources* include all resources required for performing processes that were described in the process pattern. The main type of resources that are considered in the following example are different construction machines, however also

other types like construction materials or construction teams can be described here and have to be added for a complex description.

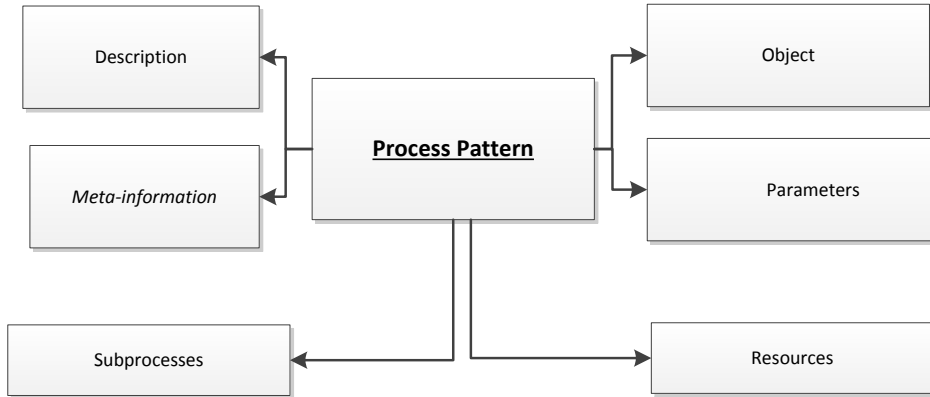


Figure 4-3: The process pattern

4.2.1. Examples

Three examples of process pattern are presented below. The first example shows the process pattern for the process of a column on-site production (cast-in-situ). The second shows the production of the precast column. In the third process pattern the production of a slab is described.

Produce a column on-site

In Figure 4-4 the structure of the process pattern "Produce a column on-site" together with its major components is shown. In Figure 4-5 the overall process is presented in the BPMN Syntax in order to show the sequence of subprocesses in it.

Process pattern describes a typical on-site production process of a column with a help of a crane. As it can be seen, the description provides a short characteristic of the process while meta-data contain some keywords for this pattern (therefore this process pattern can be easily found just by filtering all process patterns according to their production place, construction object or used construction machine). "*IfcColumn*" represents a building object in this process pattern. Two kinds of parameters for time and costs estimation are also included in this pattern, however, their exact values are not shown.

Process pattern "*Produce in situ column*" requires a crane as a resource (while other resources such as formwork, crew, materials etc. are also used in this process pattern, only machine resources will be shown, because they play a crucial role in the planning of the overall process). The pattern consists of five subprocesses, two of which can be executed in parallel, because they are independent from each other.

The subprocesses must be executed in the following order:

1. *Form* –formwork on site.
2. Can be executed in parallel:
 - Place reinforcement*– steel reinforcing bars are added to carry tensile loads.
 - Brace* – bracing formwork.
3. *Concrete* – concreting a column.
4. *Retract* – retract formwork.

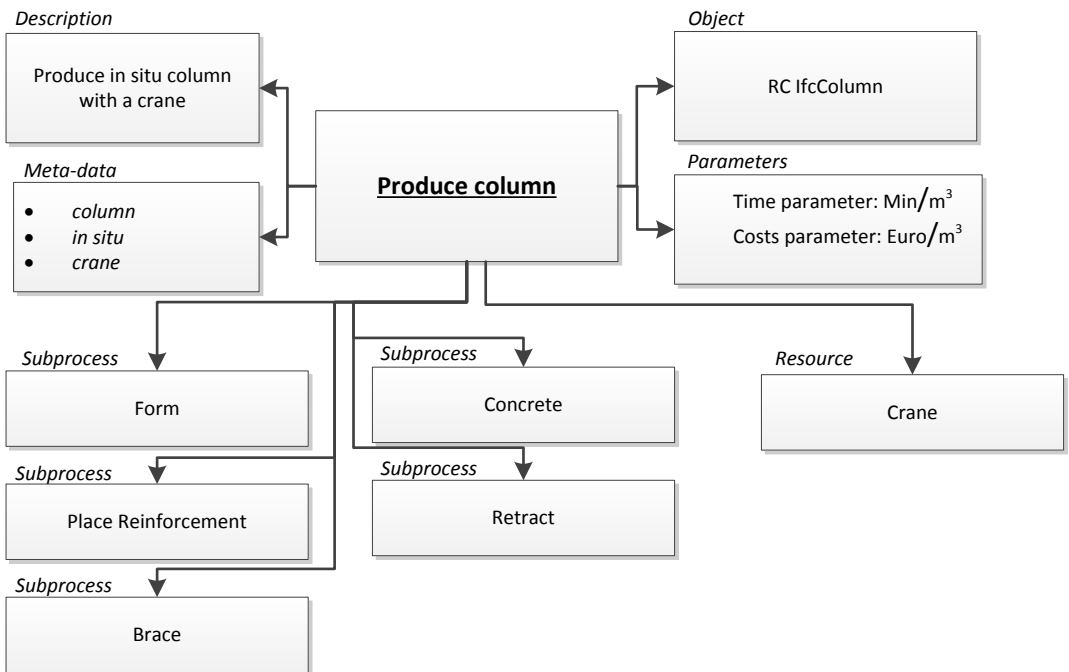


Figure 4-4: "Produce a column on-site" structure

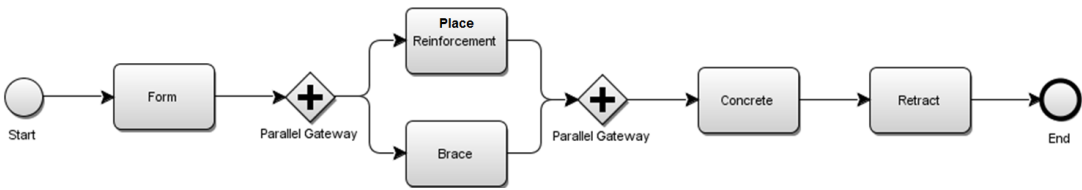


Figure 4-5: “Produce a column on-site” sequence of processes

Produce a precast column

The second example shows the process pattern for the precast column. Actually this type of column is being constructed outside of the installation place and is only brought there to be installed on place. Nevertheless this process is also called *produce column*.

The structure of the process to assemble the precast column on-site is shown in Figure 4-6. The order of the subprocesses can be seen in the BPMN representation of the process in Figure 4-7.

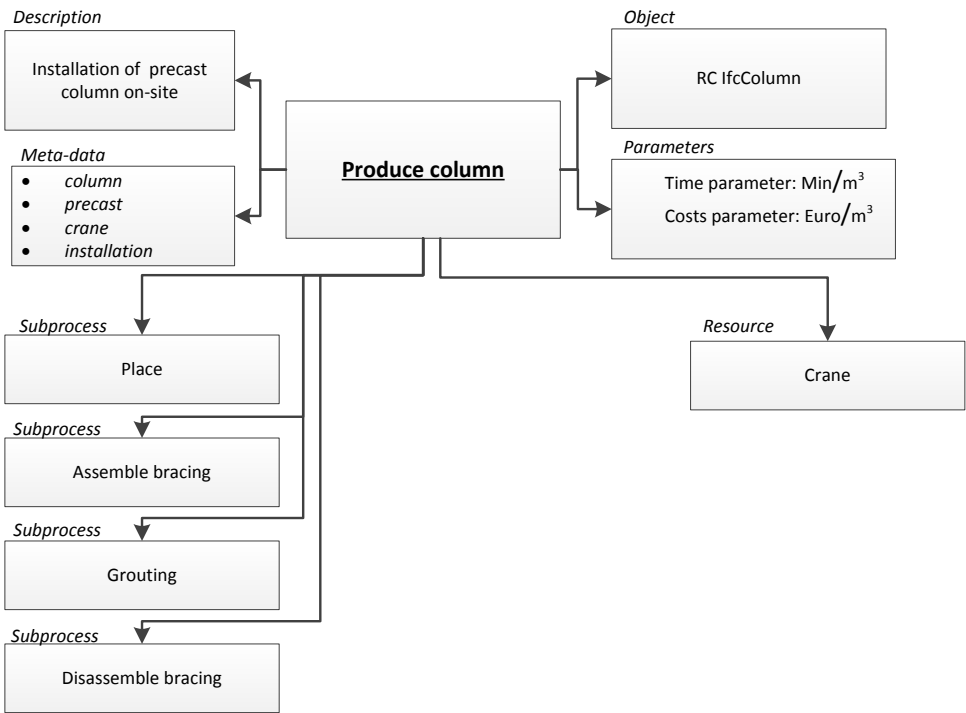


Figure 4-6: “Produce precast column” structure

The main elements of the process pattern “Produce precast column” are similar to the elements of the pattern shown before. It also has description, object, parameters and meta-data as these elements are common for all process patterns proposed in this work.

Process pattern also requires crane as a resource and consists of four subprocesses, which are executed one after the other. The subprocesses must be executed in the following order:

1. *Place* – place pre-cast column.
2. *Assemble bracing* – bracing is assembled.
3. *Grouting* – grouting precast column.
4. *Disassemble bracing* – bracing is disassembled.

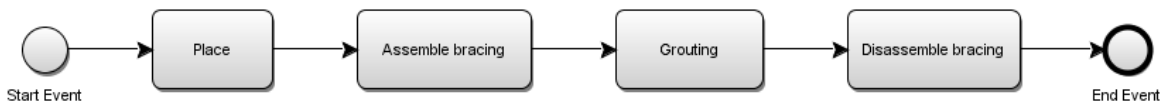


Figure 4-7: “Produce precast column” sequence of processes

Produce a slab

The last example shows the process pattern for the precast slab. The reinforced concrete slab is prefabricated outside of the installation place and is only brought there to be installed on place (see Figure 4-8).

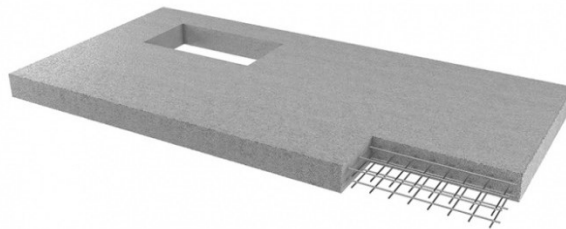


Figure 4-8: Reinforced concrete slab (Tuberías y Prefabricados Palau, S.A.¹⁹)

¹⁹ <http://www.tppalau.com> Retrieved 2015-02-11

The structure of the process is shown in Figure 4-9. The order of the subprocesses can be seen in the BPMN representation of the process in Figure 4-10.

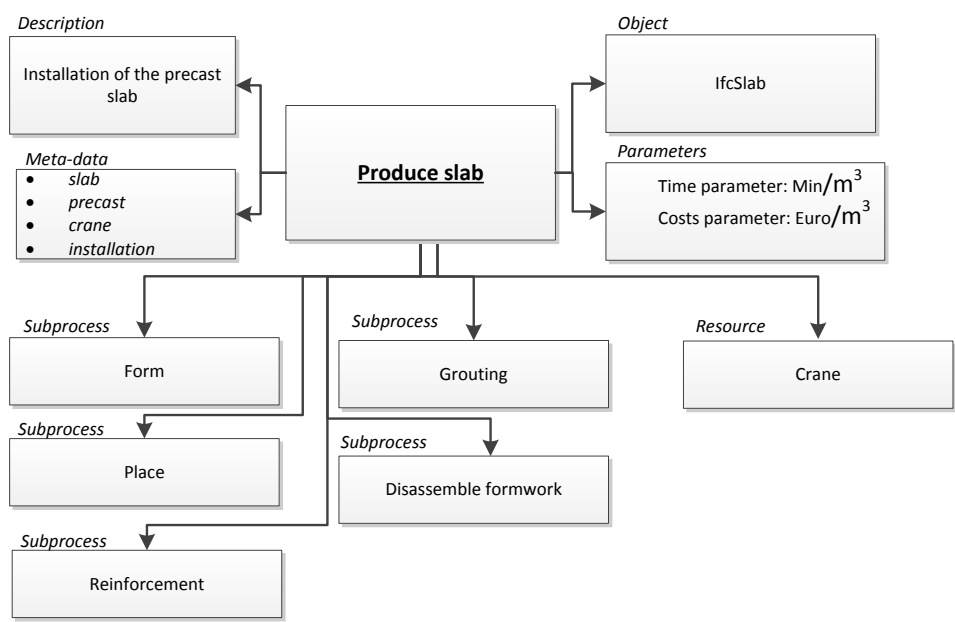


Figure 4-9: “Produce precast slab” structure

The main components of the process pattern “Produce precast slab” contain the same elements as in other patterns described above.

Process pattern requires crane as a resource and consists of five subprocesses, which are executed one after the other. The execution order of the subprocesses is following:

1. *Form* – slab boundary formwork.
2. *Place* –installation of the precast slab.
3. *Reinforcement* – assemble bound beam reinforcement.
4. *Grouting* –grouting of the precast slab.
5. *Disassemble formwork*– disassembling formwork.



Figure 4-10: “Produce precast slab” sequence of processes

4.3. Ontological framework

In order to model and store the process patterns presented in the previous section a special ontology, called Process Pattern Ontology was developed. This ontology together with the Process Instance Ontology, which was created for storing unique construction process descriptions, form a so called ontological framework used in the proposed methodology. In the following subsection structures of both ontologies as well as their interactions are considered. The description of the ontological framework starts with the discussion about the choice of an appropriate data model for representing process pattern and the comparison of two possible models: an ontology and a database.

4.3.1. Motivation for using ontologies instead of databases

Two main data models for representing knowledge in computer systems are the ontology and the database schema. They both are quite similar and have some analogous characteristics.

Database models have been actively used in last decades in many software applications as efficient and reliable data models. Ontologies as an alternative to databases have appeared recently (the first working drafts of the OWL standard were published in 2002), but they are competing nowadays in popularity with the database schema due to active development of the Semantic Web applications.

Entities in the database schema can be seen as ontology classes, attributes and relations as ontological properties, ontology instances are similar to the database data and even the database schema describing structure of the data and constraints on them is analogous to the ontology axioms. However, there are some important differences between the ontology and the database schema and therefore, in this subsection it will be explained, why processes in this work are modeled in the ontology and not in a relational database, which is also well-known, robust and established as a methodology.

At first it is important to define the core purposes of both data models. Databases are focused primarily on data and their main purpose is to structure data for efficient storage and querying. Ontologies are focused not only on data, but also on their semantics. Of course the semantics of ontology depends on the used representation language, but for all realizations it is significantly richer than in databases.

Ontological representation mixes the schema specification with instances; however it is not obligatory that ontology has instances. Furthermore the closed world assumption is applied to databases, which means that missing information is treated as false. While in the open world assumption, which is applicable to ontologies, the missing information is treated as unknown.

The backbone of ontologies is a hierarchical structure of the concept. Therefore, it is much easier to present the complex hierarchical structure of the processes in ontology using its taxonomical concept inclusions. On the other hand, the main idea here is not to purely structure the data in order to put queries as it is useful in a database, but to have a flexible structure, which can be easily changed and adopted. For example, to model a construction process related to the one, which exists in the ontology the corresponding concept inclusion and some new roles can be added. This can be very useful if new types of building elements or construction machines are added to the ontology. Meanwhile in a database it can be the case that the whole table structure should be reviewed.

Searching for the data is also a bit more flexible in ontology. The user does not need to know the exact context of the tables or to use fixed predefined queries if he works with ontology. Therefore it is possible to search for a process pattern without specifying all parameters of the query, in case we do not know which resources or construction machines are available. Due to this feature of ontology together with the platform-independency the system becomes very interoperable, what is important in huge construction projects with many participating partners.

Instead of the "*data integrity*" used in databases, the "*consistency check*" can be performed. Another option, which can be copiously realized in databases by means of entering the "*null*"-value in the table field, is to model incomplete information in the ontology, with the possibility to add it later, what does not lead to a mistake. This is actually very often the case, as at the beginning not all information related to the construction project is available and saved in the ontology.

One more feature of ontology is a rich possibility to present the knowledge with its complex semantics and to reason on it. It allows deducing new information by reasoning on some facts in ontology. In a database semantics is not explicitly expressed. Moreover, the structure of the tables in a database can be so complex, that it can be very time consuming to find the meaning of fields dependencies. One of the good illustrations for this difference is a sophisticated representation of transitivity in databases, while in ontology this is quite easy.

It is also important to note that there are several approaches for generating databases schemas from ontologies as it was proposed in (Vysniauskas & Nemuraite, 2006) and (Gali, et al., 2005). However they all have a major disadvantage. Always a lot of ontology related features, as well as some semantics are lost in the transformation process.

Open Word and Closed Word Assumption

Discussing about ontology and other data models it is important to take a look at two assumptions that are used in logic for the knowledge representation. These are the Closed World Assumption (CWA) and the Open World Assumption (OWA).

- The Closed Word Assumption is the assumption that if some fact is not known (to be true), it must be false.
- The Open World Assumption is the assumption that if some fact is not known (to be true) it is just unknown.

The CWA and the OWA are two very particular ways of modeling providing different views of the world. In the CWA the information of a system is assumed to be complete and therefore no explicit declaration of the falsehood is needed. The CWA is more commonly used especially in the case of database applications. In contrast to the CWA in the OWA the unknown information is not considered to be false automatically and therefore the OWA is applied to the system which has incomplete information. The use of the OWA is a good choice in case when the information from different sources or models (sometimes incomplete) is combined.

Following benefits of the open world framework for knowledge management application in the enterprise were summarized in (Bergman, 2009):

- Domains can be analyzed and inspected incrementally;
- Schema can be incomplete and developed and refined incrementally;
- The data and the structures within these open world frameworks can be used and expressed in a piecemeal or incomplete manner;
- It is possible to combine data with partial characterizations with other data having complete characterizations;
- Systems built with open world frameworks are flexible and robust; as new information or structure is gained, it can be incorporated without negating the information already resident;
- Open world systems can embrace closed world subsystems.

However it is important to add that in the case when the domain is well-characterized, schema is not very large and the complete information is available a traditional relational model with the CWA should be used.

Comparison

A short comparison of ontology and database schema is shown in Table 6.

Table 6: A comparison of ontology language and database schema

| | Ontology | Database |
|------------------|---|--|
| Focus | Semantics | Data |
| Similar elements | Entities | Classes |
| | Attributes | Properties |
| | Relations | |
| Expressivity | Taxonomy and classes hierarchy are backbone | No taxonomy, only table structure |
| Search | SPARQL | SQL |
| World assumption | Open world assumption | Closed world assumption |
| Instances | Instances are optional | Instances are essential |
| Performance | No efficient ontology management systems nowadays | Very robust, efficient database management systems |

Limitations

One of the main considerations by developing an ontology is the complexity of the ontology engineering process. The ontology engineering is a set of activities that concerns the ontology development process, the ontology life cycle, the methods and methodologies for building ontologies, and the tool suites and languages that support them (Gomez-Pérez, et al., 2004). Because of the fact that an ontology tries to represent the complex semantics of concepts and relations among them, as well as properties, axioms and rules, its schema is quite complex and therefore it also requires much more time to be developed.

By using the ontological approach it definitely has to be taken into account that such approaches are still in development, and therefore ontologies are not as robust as databases. Especially some problems can arise while working with ontologies having a lot of

instances. Ontologies are often represented in plain OWL files and do not have so far such an efficient mechanism for managing instances as many database management systems can provide nowadays. For example a building information model of an airport terminal can consist of several thousands or millions of elements and relations between them. A possible solution is to store ontologies in a database environment. This should not be confused with the approach of transformation of ontologies to the databases schemas mentioned above. This approach means that a complete ontology structure with its semantics and properties is stored in a relational database. A Jena2 Database Interface²⁰ provides an application programming interface for storing ontologies in the most common database engines (such as MySQL, Oracle DB or Microsoft SQL Server). RDF data are stored in a relational database, however, the database layout used by Jena is not intended for direct access by users and applications and the ontology should be accessed only indirectly through the Jena API.

A hybrid solution would be to store the part of the information, e.g. mass data from sensors in relational databases, while the other part, e.g. system information would be stored in ontology, in order to be able to provide flexibility, reasoning and extension capabilities for the system (Faschingbauer, 2011).

Also it is necessary to take care about data access and safety handling by ontologies. However, due to a lot of advantages mentioned above, hierarchical structure for the modeling of concepts, support of the semantic search mechanism as well as its reasoning capabilities, ontology has been chosen for modelling construction processes.

²⁰ <http://jena.sourceforge.net/DB> Retrieved 2013-03-21

4.3.2. Ontology for process patterns

The Process Pattern Ontology (PPO) plays a superior role in the developed architecture. It provides reusable process patterns that can further serve for dynamic process configuration.

Construction processes can be classified into information processes and material processes. Information processes are associated with planning and controlling procedures with their requirements and corresponding objects. Material processes usually take place on the construction site and are associated with the production and logistics processes. In the scope of this research we concentrate our efforts on material processes (namely production processes), however, similar structure and approach also can be used for information processes.

The two main upper level concepts in the Process Pattern Ontology are *Process* (Figure 4-11) (4.1) and *Object* (4.2) (Figure 4-12).

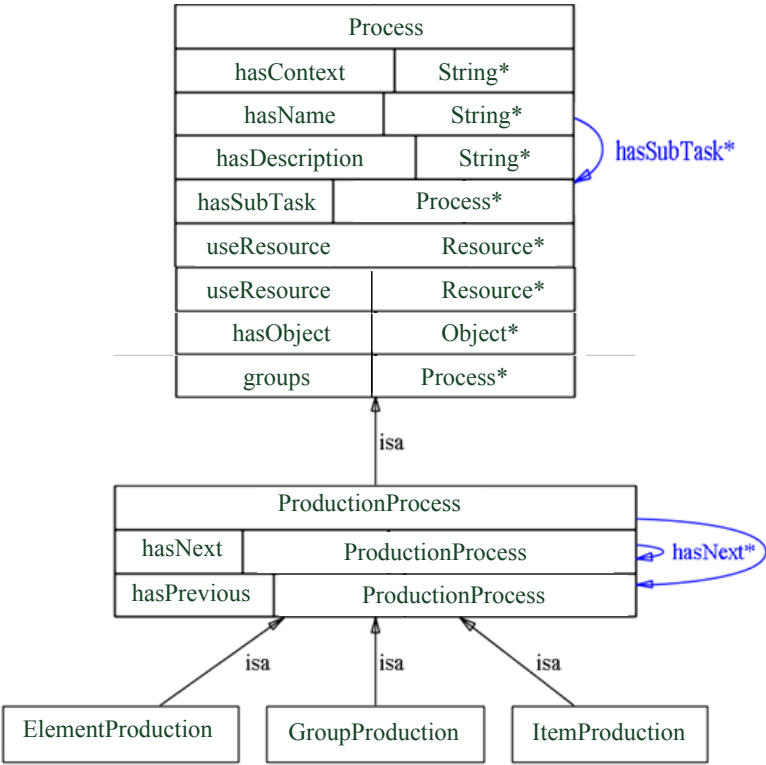


Figure 4-11: Process concept with major properties and inter-relations

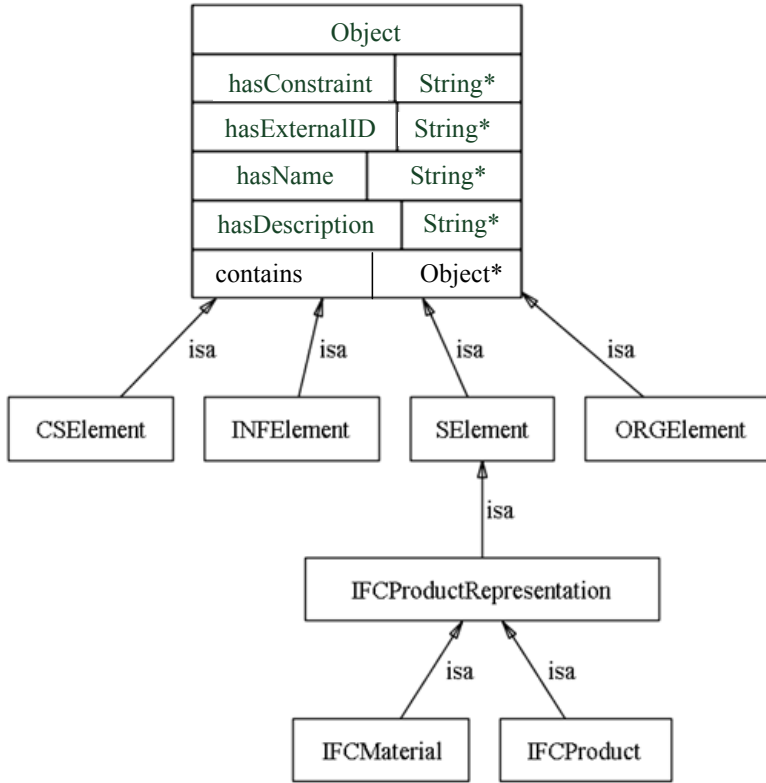


Figure 4-12: Object concept with major properties and inter-relations

Process is a base concept to model all kinds of production processes that are considered in our work.

$$Process \sqsubseteq \top \quad (4.1)$$

$$Object \sqsubseteq \top \quad (4.2)$$

In OWL two types of properties (relations) are defined:

Object properties – link individuals to individuals (\cong Relation);

Data properties – link individuals to data values (\cong Instantiation).

In order to build a process hierarchy, an object property *hasSubTask* (4.3) can be used. Other object properties are used to create a process workflow (*hasNext* (4.4) and *hasPrevious* (4.5)) or to reference a certain Object (*hasObject* (4.6)).

$$\forall hasSubTask. Process \sqsubseteq Process \quad (4.3)$$

$$\forall hasNext. Process \sqsubseteq Process \quad (4.4)$$

$$\forall hasPrevious.Process \sqsubseteq Process \quad (4.5)$$

$$\forall hasObject.Object \sqsubseteq Process \quad (4.6)$$

Three subconcepts of the production process are defined in the ontology. These concepts that inherit all properties of a higher-level concept are:

- *GroupProduction* (4.8) – is used to model processes producing group of elements and composed of several *ElementProduction* processes (e.g. *Produce_Walls* or *Produce_Columns*).
- *ElementProduction* (4.9) – models a production process of an element and comprises several *ItemProduction* processes (e.g. *Storey_produce*, *Slab_produce*, *Wall_produce* or *Column_Produce*).
- *ItemProduction* (4.10) – defines very specific elementary subprocesses which cannot further contain smaller subprocesses (e.g. *Form*, *Brace*, *Concrete* or *Re-tract*).

$$ProductionProcess \sqsubseteq Process \quad (4.7)$$

$$GroupProduction \sqsubseteq ProductionProcess \quad (4.8)$$

$$ElementProduction \sqsubseteq ProductionProcess \quad (4.9)$$

$$ItemProduction \sqsubseteq ProductionProcess \quad (4.10)$$

The *Object* concept models different kinds of objects that are used by the production processes and contained in other construction data models. The *Object* concept consists of following four subconcepts representing respective groups of objects: construction site elements (4.11), informational elements (4.12), structural elements (4.13) and organizational elements (4.14). The *IFCObject* concept, for describing different construction elements, is modelled as a subclass of structural elements (4.15).

$$CSElement \sqsubseteq Object \quad (4.11)$$

$$INFElement \sqsubseteq Object \quad (4.12)$$

$$SElement \sqsubseteq Object \quad (4.13)$$

$$ORGElement \sqsubseteq Object \quad (4.14)$$

Object property *groups* (4.16) (inverse of *hasGroup* (4.17)) is defined in order to model relations between complex objects.

$$IFCObject \sqsubseteq SElement \quad (4.15)$$

$$\exists groups. IFCObject \sqsubseteq IFCObject \quad (4.16)$$

$$groups \equiv hasGroup^- \quad (4.17)$$

With the help of the ontology it is possible to define the hierarchy of concepts, where subconcepts inherit all properties of their superconcepts. This allows building a very flexible and generic structure for the process pattern. For example by modelling the upper concept "*machine*" with a few subclasses (4.19) (e.g. "*crane*", "*pump*" etc.), a process using a specific construction machine references to the upper concept, but not to the subconcepts (4.20). Therefore, the process pattern can also be used even when the detailed information about the machine type is missing. In that case the generic concept "*machine*" will be used during the instantiation and later an additional relation defining a type of the machine can be added.

$$Crane \sqcap Pump \equiv \perp \quad (4.18)$$

$$Machine \sqsubseteq Crane \sqcup Pump \sqcup \dots \quad (4.19)$$

$$\exists useResource. Machine \sqsubseteq Process \quad (4.20)$$

An example of such a pattern describing the process of column production with a crane (this pattern was described in section 4.2 *Produce a column on-site*) is shown in Figure 4-13.

An ontological representation of the process pattern in OWL/RDF serialization is presented in Figure 4-14.

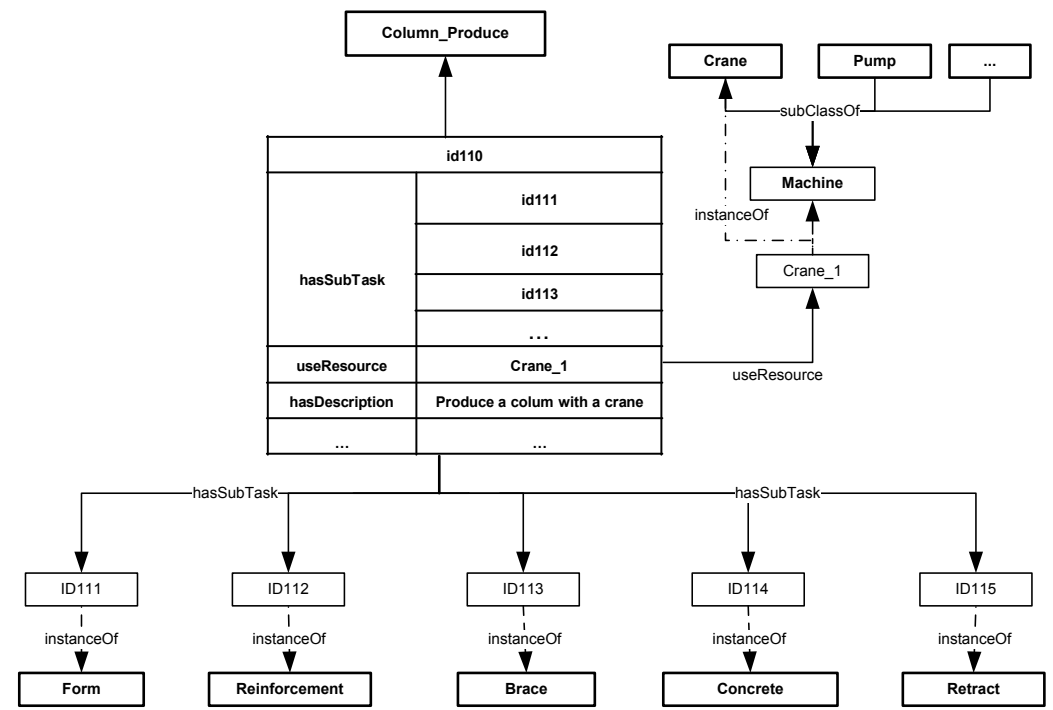


Figure 4-13: Pattern of the process „Column produce with a crane“

```

<owl:Class rdf:about="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Column_produce">
  <rdfs:subClassOf rdf:about="http://www.mefisto-
    bau.de/ontologies/ProcessOntology.owl#ElementProduction"/>
</owl:Class>

<owl:Class rdf:about="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Crane">
  <rdfs:subClassOf rdf:resource="http://www.mefisto-
    bau.de/ontologies/ProcessOntology.owl#Machine"/>
</owl:Class>

<Column_produce rdf:about="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#id110">
  <hasDescriptionrdf.datatype="http://www.w3.org/2001/XMLSchema#string">Produce column with
    crane
  </hasDescription>
  <hasContext rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Column</hasContext>
  <hasContext rdf:datatype="http://www.w3.org/2001/XMLSchema#string">crane</hasContext>
  <hasContext rdf:datatype="http://www.w3.org/2001/XMLSchema#string">produce</hasContext>
  <hasSubTask>
    <Brace rdf:about="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#id114">
      <hasDescriptionrdf.datatype="http://www.w3.org/2001/XMLSchema#string">
        Brace column</hasDescription>
    </Brace>
  </hasSubTask>
  <hasSubTask>
    <Concrete rdf:about="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#id115">
      <hasDescriptionrdf.datatype="http://www.w3.org/2001/XMLSchema#string">
        Concrete column with crane</hasDescription>
    </Concrete>
  </hasSubTask>
  <hasSubTask>
    <Retract rdf:about="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#id116">
      <hasDescriptionrdf.datatype="http://www.w3.org/2001/XMLSchema#string">
        Retract column</hasDescription>
    </Retract>
  </hasSubTask>
  <hasSubTask>
    <Form rdf:about="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#id111">
      <hasDescriptionrdf.datatype="http://www.w3.org/2001/XMLSchema#string">
        Form column</hasDescription>
    </Form>
  </hasSubTask>
  <hasSubTask>
    <Reinforcementrdf:about="http://www.mefisto-
      bau.de/ontologies/ProcessOntology.owl#id112">
      <hasDescriptionrdf.datatype="http://www.w3.org/2001/XMLSchema#string">
        Reinforcement works</hasDescription>
    </Reinforcement>
  </hasSubTask>
  <useResource>
    <Crane rdf:about="http://www.mefisto-
      bau.de/ontologies/ProcessOntology.owl#Crane_1"/>
  </useResource>
</Column_produce>

```

Figure 4-14: Process pattern in OWL/RDF serialization (presented partially and without sequence relations)

4.3.3. Ontology for process instances

In comparison with the Process Pattern Ontology, the content of the Process Instance Ontology (PIO) is unique for each modeled construction process. It stores concrete process descriptions and is uniquely populated with specific process assertions for each construction case.

The Process Instance Ontology has the same class taxonomy as the Process Pattern Ontology, however, its set of defined properties is extended by some additional data properties used for more qualitative and quantitative description of the process to become a workflow.

For the concept *Object* following data properties are defined (corresponding data types of the properties are shown in formulas (4.1-8) in braces):

- *hasExternalID* (4.21) – is used to link objects in the ontology with their representation in other models (for example in IFC model).
- *hasX* (4.22), *hasY* (4.23), *hasZ* (4.24) – are used to assign geometrical information about the coordinates of the object in the building.
- *hasNumber* (4.25)– is used to assign a number to a storey element of a building.

$$\forall \text{hasExternalID}. \{data\ type: String\} \sqsubseteq Object \quad (4.21)$$

$$\forall \text{hasX}. \{data\ type: Double\} \sqsubseteq Object \quad (4.22)$$

$$\forall \text{hasY}. \{data\ type: Double\} \sqsubseteq Object \quad (4.23)$$

$$\forall \text{hasZ}. \{data\ type: Double\} \sqsubseteq Object \quad (4.24)$$

$$\forall \text{hasNumber}. \{data\ type: Integer\} \sqsubseteq IFCStorey \quad (4.25)$$

For the *Process* following data properties were added:

- *hasPriority* (4.26) – is used to assign a priority value for the processes. The priority values are used in some configuration approaches for construction processes.
- *hasDuration* (4.27) – is used to assign an approximate duration value to a process that can be calculated with the help of time parameters.
- *hasCost* (4.28)– is used to assign an estimated cost value to a process that can be calculated with the help of cost parameters.

$$\forall \text{hasPriority}. \{data\ type: Integer\} \sqsubseteq Process \quad (4.26)$$

$$\forall \text{hasDuration}. \{data\ type: String\} \sqsubseteq Process \quad (4.27)$$

$$\forall hasCost.\{data\ type:String\} \sqsubseteq Process \quad (4.28)$$

Initially this ontology has no instances; they are added sequentially at runtime during the configuration process. The instantiation process of this ontology is semi-automatic and cannot be fully automatic because of the fact that, in certain cases, it is necessary to set concrete object identifiers in predefined pattern structures from the Process Pattern Ontology. Many identifiers could be generated automatically; however, references to a specific construction object or construction machine have to be assigned manually in some cases.

While the Process Pattern Ontology cannot be modified by the end user and serves only for the pattern search and retrieval, the Process Instance Ontology is constantly changing by adding new instances or relations between them.

The class taxonomy of the Process Instance Ontology is shown partially in two parts in Figure 4-15 (a taxonomy for concept Object) and Figure 4-16 (a taxonomy for concept Process).

In order to interact with other domain models corresponding concepts were defined in the Process Pattern and Process Instance Ontologies. Moreover, for the case of the standard BIM the structure of these concepts were kept similar to the original ones defined in IFC (ISO PAS 16739) data model; therefore in the ontologies there are concepts like *IfcBuilding*, *IfcStorey*, *IfcMaterial*, *IfcColumn* and others. By querying the specific data models it is possible not only to instantiate these concepts but also obtain various additional topological and structural dependencies, such as: "All *IfcStorey* elements in an *IfcBuilding*", "How many *IfcColumn* elements are in one specific *IfcStorey*", "Which *IfcElements* are between axes A and B on *IfcStorey X*" etc.

A possible extension of the ontological framework with the *ifcOWL*, an ontology for building and construction sector based on the industry foundation classes is discussed in the outlook section of chapter 8.

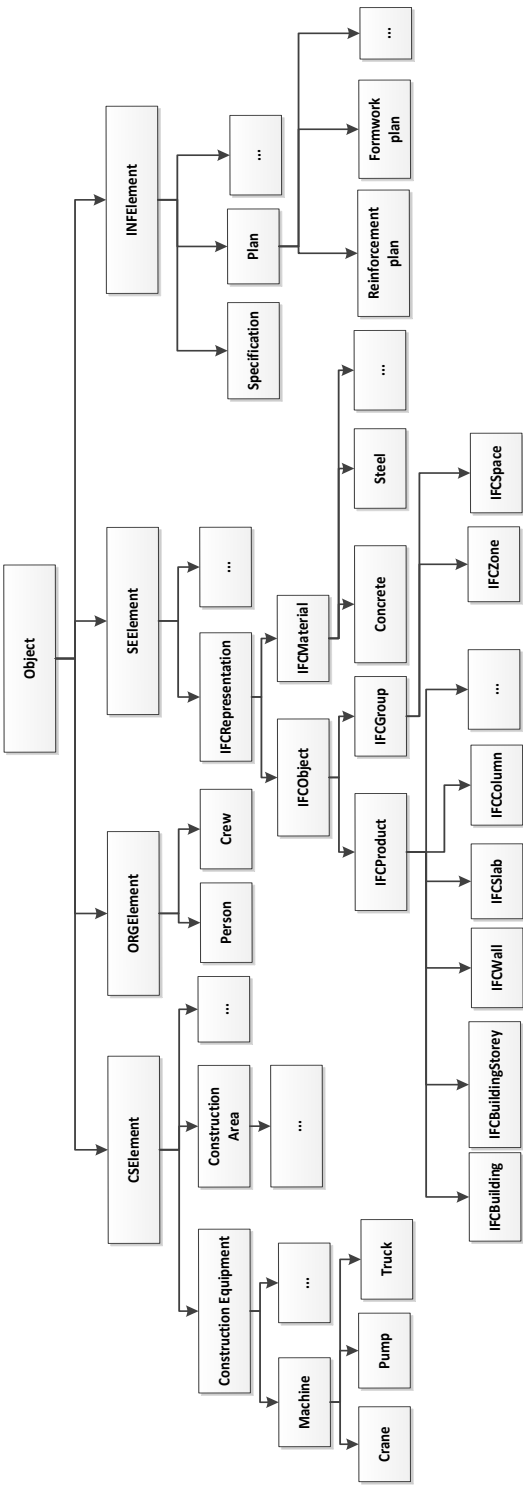


Figure 4-15: A taxonomy for the concept *Object*

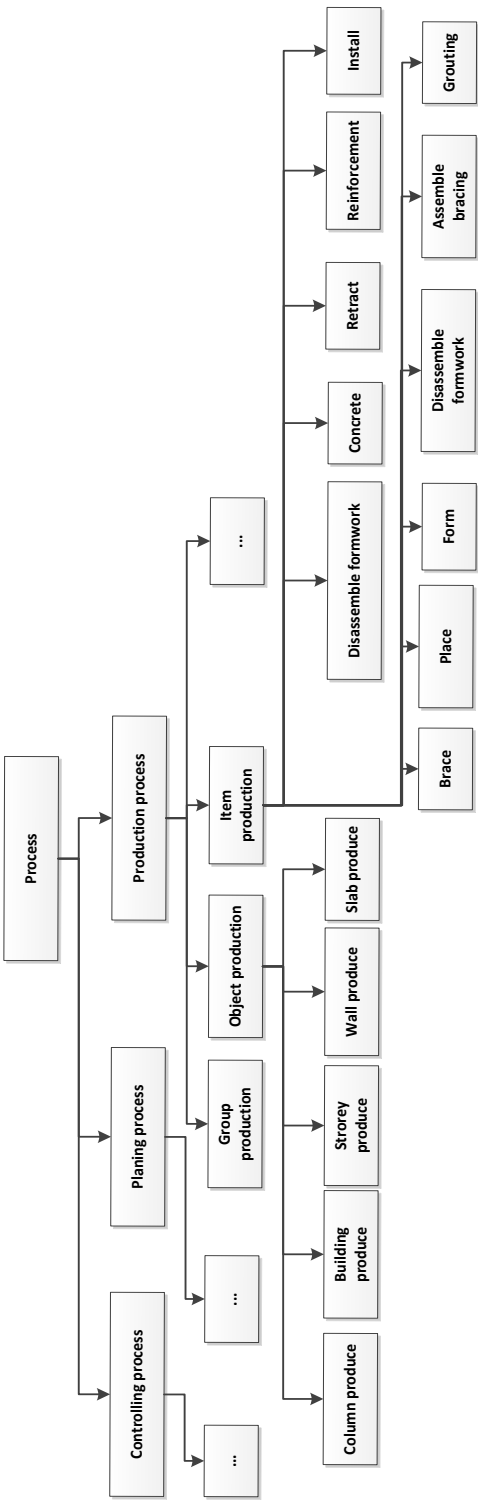


Figure 4-16: A taxonomy for the concept *Process*

5. Configuration of construction processes

Process patterns introduced in the previous chapter provide a general structure for describing typical construction processes. In this chapter an approach for the configuration of construction processes by applying process patterns is considered and developed. It includes three main steps that are discussed in the following sections.

In the first section the generation and configuration approach and its main steps are presented.

The second section introduces the mechanism of the process pattern retrieval and demonstrates the extraction of the required process pattern from the Process Pattern Ontology with the help of the RDF query language SPARQL.

In the third section the process adaptation method is explained. The adaptation is an intermediate step that is required to adapt the generic process patterns to meet the specific project requirements and instantiate it.

In the final section the configuration method of process modules, obtained after the adaptation step is outlined. There the concept of configuration rules is presented and a comparison between different configuration approaches and rule-engines is carried out, which can be applied for local optimization.

The last part of this section is dedicated to the configuration strategies developed and implemented in this work, which are necessary in order to perform global optimization.

5.1. Generation and configuration

The approach for the generation and configuration of construction processes developed in this work consists of the following three main steps (s. Figure 5-1):

- Retrieval;
- Adaptation;
- Configuration.

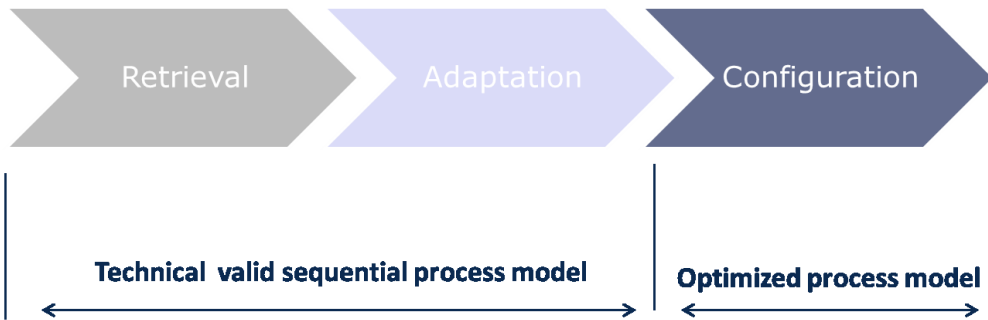


Figure 5-1 Generation and configuration steps

At the first step the required process patterns are retrieved from the Process Pattern Ontology. These patterns describe process on a very general level and therefore they should be adapted to the specific project case at the second step.

After the pattern retrieval and adaptation a technical valid sequential process model is stored in the Process Instance Ontology. The third configuration is an optional step, because after the first two steps a valid process model is already generated. However this model can be further configured with the help of various configuration rules and strategies in order to get an optimized process model.

5.2. Process pattern retrieval

The problem of finding a required process pattern is considered here. The Process Pattern Ontology described in section 4.3 provides an efficient data storage for reusable process patterns. An important problem is how to support the user in finding the right or the most suitable process pattern. This includes two tasks: a mechanism of process pattern retrieval, describing the extracting of the selected pattern from the ontology and the adaptation of the process pattern to specific project requirements.

The ontological structure of the data model and additionally added meta-data information allow searching for all patterns satisfying requirements and then filtering them using certain conditions. Each pattern is annotated with specific meta-data information and therefore two kinds of searches can be performed:

- The first one is a keyword search, which allows searching for by using specific keywords from the process patterns meta-information section (like “crane” or “column”). This search mechanism does not allow defining very complex query

structure and therefore it should be used only optionally. For example it is not possible with the keyword search to formulate a query with few various conditions and variables.

- The second search mechanism can be used to create more sophisticated queries, including different operators and variables. It uses a set of triple patterns called a basic graph pattern and is described in the following section.

Search with graph pattern

Information in both ontologies is stored in the form of directed and labeled OWL/RDF graphs and therefore a SPARQL query language is used to retrieve and manipulate data stored in these models.

As a query language SPARQL has different operators to restrict the solutions of a graph pattern match according to a given expression, filter them or modify the solutions after patter matching. The variables in the search graph pattern are indicated by a " ? ". The results of the execution of the query are results sets or OWL/RDF subgraphs, where all variables are bound.

The SPARQL query processor searches in the Process Pattern Ontology for the set of triples that matches the graph pattern. The pattern retrieval process is shown schematically in Figure 5-2. It demonstrates a graph pattern with one variable, different triples available in the Process Pattern Ontology and a founded solution in which search variable is bound to *B*.

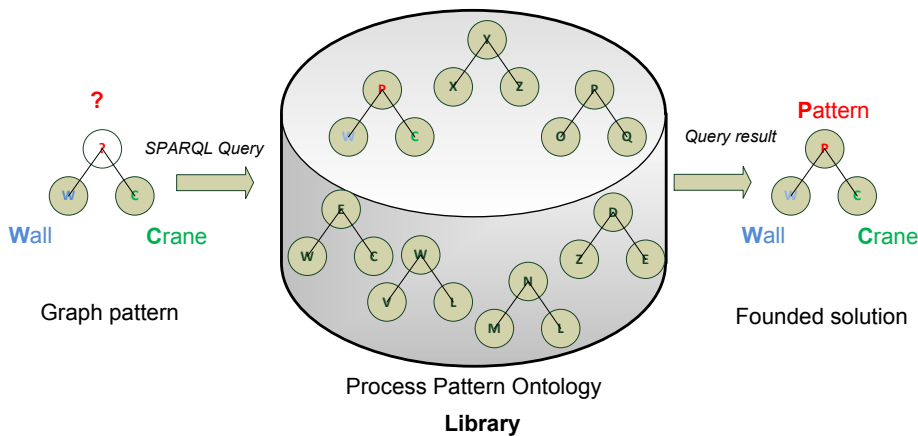


Figure 5-2 Process pattern retrieval

Ontological structure allows performing different and quite comprehensive types of queries. For example it is possible to search for a pattern that constructs some structural

element by using a specific construction machine. Moreover it is not necessary to specify the type of a construction machine in each case. This is the case, when the information about the available resources is missing at the time of the query. An example for such a query is: *"Find all patterns that construct a column with some machine"*.

Below in Figure 5-3 an example of a query is shown, where the type of the construction machine can be defined explicitly: *"Find all patterns that construct an object with a construction machine"*. As already mentioned, to query the Process Pattern Ontology the SPARQL query language is used. SPARQL support is available in the Jena framework, which allows executing queries with the Jena API and integrating them in any Java application. A simplified SPARQL query for the example mentioned above is presented below.

The example query uses three variables:

- *Variable1* defines the type of the object ("column" or "wall"),
- *Variable2* defines the type of the construction machine (can be a generic "machine" or specific classes "crane" or "pump"),
- *Variable3* specify the type of the task ("produce" or "concrete").

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mf: <http://www.mefisto-bau.de/ontologies/ProcessPatternOntology.owl#>
SELECT ?uri
WHERE {
    ?uri mf:hasContext ?context.
    ?uri mf:useResource ?resource.
    ?uri mf:hasObject ?object.
    ?object rdf:type Variable1
    ?resource rdf:type Variable2
    FILTER (REGEX(STR(?context), Variable3)) }
}
```

Figure 5-3: An example query

5.3. Process adaptation

Process patterns stored in the Process Pattern Ontology describe typical construction processes with their resources and subprocesses on a very general level (they specify domain classes and not particular instances) and therefore they cannot be directly copied in the Process Instance Ontology after the retrieval. Process patterns have to be adapted and instantiated to suit specific project conditions of particular process modules from the Process Instance Ontology. These instantiation and modification are done in an additional intermediate *process adaptation step* taking place directly after the retrieval process.

Each process pattern contains much more information than is available in the Process Instance Ontology and therefore in this step some new elements are generated and added to the ontology. In most cases at that time only information about the building objects, available resources and some processes is saved in the Process Instance Ontology. Thus in the *process adaptation step* all subprocesses with their sequence relations have to be generated and added to the Process Instance Ontology. Moreover links between processes and required resources are being established and the information about the used process pattern is saved in the instantiated process modules. Storing information about the used process pattern in the instantiated modules has a major benefit, that in case of some changes in the process pattern in the future, these modifications can be easily propagated to all instantiations of the pattern. In Figure 5-4 an adaptation step for the process pattern "*Produce a precast column*" is shown. Process pattern and process instance are combined together and a new process modules appears in the Process Instance Ontology.

The process adaptation step consists of several substeps, which can be also recursive:

- Process adaptation;
- Subtasks adaptation;
- Resource adaptation;
- References generation.

At the beginning, after the process pattern is selected, the area of its application is defined. Process pattern can be applied to some selected elements or to all elements of one group. The adaptation process begins with the instantiation of the main process and its related parameters. After that the related subtasks are instantiated and relevant links to the main process are generated. Then the resources are linked to the respective processes. The adaptation and instantiation process consists of several methods and cycles and is implemented in the Java programming language by using special APIs to work with ontologies. These cycles can be also recursive and has its goal to go through the whole structure of pattern (similar as in the branch and bound search algorithm) and

to link retrieved process pattern with the raw process instance and required resources in order to generate the detailed process instance (s. Figure 5-4).

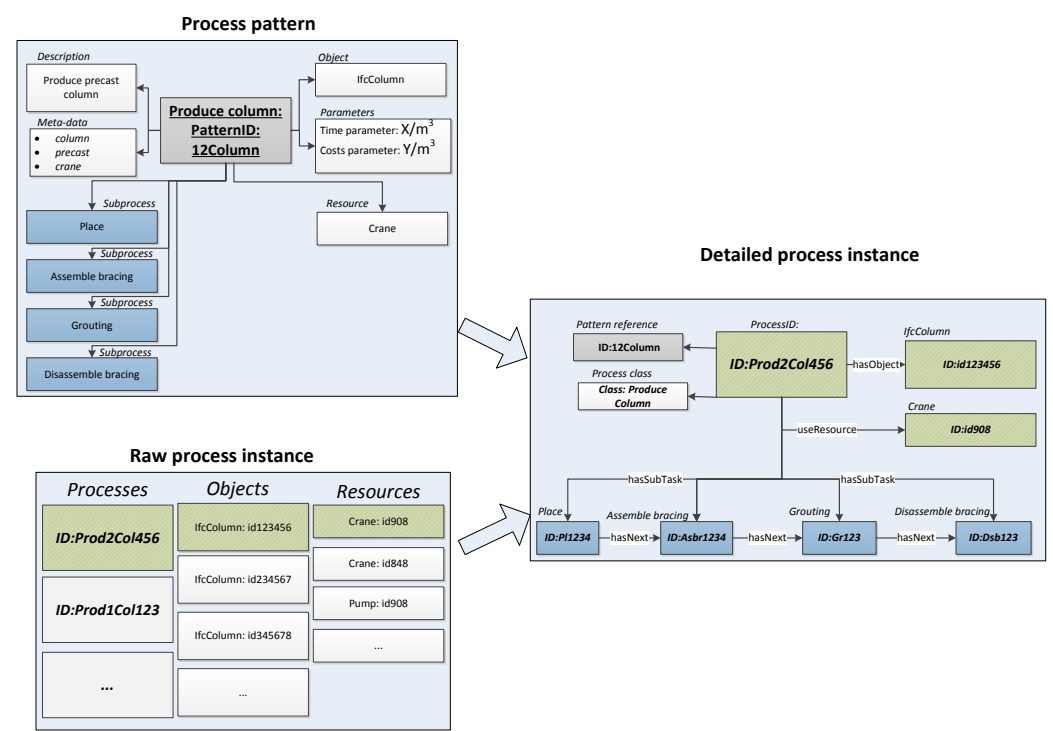


Figure 5-4: Adaptation step

5.4. Process configuration

Process modules stored in the Process Instance Ontology after the adaptation step contain all subprocesses and are linked to the required resources. By default they are configured to be produced sequentially if not otherwise stated in the pattern explicitly because this is always a valid solution. Other better solutions are not always valid and therefore cannot be used for the default configuration.

The sequential execution of the processes is not always an optimal and rational production order. Some processes can be executed in parallel, others can be grouped and prioritized depending on different conditions that are unique for each construction project and, therefore, cannot be considered and modelled optimally in advance at the level of process patterns. This can be done at the third configuration step at which an optimized and valid process model is generated.

5.4.1. Configuration rules

Process configuration with rules is an optional step, because even after the adaptation step, the complete process model with all necessary relations is already stored in the Process Instance Ontology. However, this model can be modified and restructured in the configuration step with the help of various configuration strategies in order to improve its quality and make the overall process more time- and cost-efficient.

Due to the use of the ontological structure for the formalization of the construction process there is a possibility to apply a reasoning mechanism in order to analyze or modify a configured process. Using rules allows us to set and then check some specific construction constraints, and perform further the configuration of the processes saved in the Process Instance Ontology.

Configuration rules considered in this work are similar to the if-then statement of traditional program languages and are of the form (5.1):

$$\text{IF some conditions THEN some actions} \quad (5.1)$$

The *if* part of the rule is often called its *left-hand side* (shortly *LHS*), premises or predicate and the *then* part is called the *right-hand side* respectively.

In the formal logic syntax the rule can be written as (5.2):

$$B_1, B_2, \dots, B_n \rightarrow H \quad n \geq 0 \quad (5.2)$$

where H is the head(sometimes also called conclusion, action or *RHS*) and B_1, B_2, \dots, B_n are body of the rule (also called premises or *LHS*). H, B_1, B_2, \dots, B_n are some syntactic constructs that include variables and different logical operators. Set of rules are usually called a rule-base.

Rule-based systems are systems that use rules to derive conclusion from premises. A rule-base together with an inference engine and a working memory are the main components of a rule-based system. One of the main applications of rule-based systems can be found in the area of the artificial intelligence where they are used in the expert systems, computer systems emulating the decision-making process of a human expert. Using rule-based systems together with ontologies has an advantage that we can do reasoning not only about the value of the objects but also about their structure and semantic relations between them.

The rule-based system for process configuration with ontology consists of three components (Figure 5-5):

- Knowledge base (Fact base), in our case – ontology;
- Set of rules (Rule base);
- Control system with a rule interpreter (Inference engine).

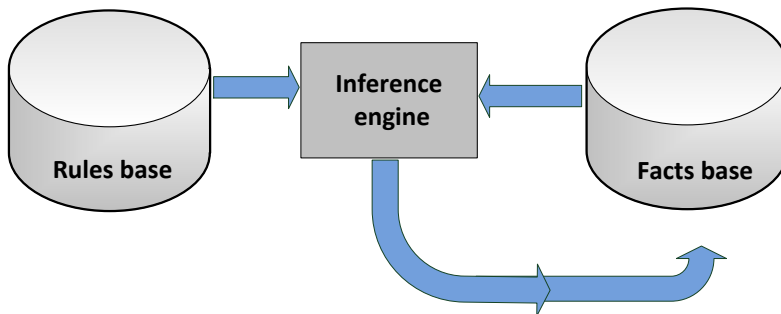


Figure 5-5: Rule-based system for process configuration

Currently, the integration of ontologies and rules is a subject of active research. The existing approaches can be assigned to two categories (REWERSE, 2008):

- *Hybrid approach*: there is a strict differentiation between ontology predicates and rule predicates. Reasoning in such systems can be done by combining of the existing rule engine with the ontology. In this approach very complex rule constructs can be defined.
- *Homogenous approach*: both ontologies and rules are represented in the same logical language, it makes no distinction between rule predicates and ontology predicates. In order to reason a general reasoner is used. The use of a homoge-

nous approach has the major advantage that an additional integration of the rule-engine with ontology is not necessary.

For the technical implementation of the process configuration with rules two different approaches were used in this work. The use of the homogenous approach was realized with the help of the Jena rules (Benevolenskiy, et al., 2014). The hybrid approach was tested with the use of the Drools rule engine (Benevolenskiy, et al., 2012).

5.4.2. Homogenous approach

Different technical implementations of the homogenous approach are presented below.

Semantic Web Rule Language

The use of Semantic Web Rule Language (W3C Working Group, 2004) is an example of the homogenous approach in the integration of ontologies and rules.

Semantic Web Rule Language (SWRL) is a rule language proposed for the Semantic Web that is based on combination of the OWL DL and OWL Lite sublanguages with a subset of the Rule Markup Language. SWRL is an expressive OWL-based rule language that can use the vocabulary of the ontology and allows writing rules to reason about OWL individuals and to infer new knowledge from them. Therefore no additional transformations between ontological knowledge base and the rule-base are required. That was one of the reasons why the initial consideration was to use the Semantic Web Rule Language (SWRL) as a rule language for the proposed methodology.

Few types of syntax are defined for the SWRL rules. Two most well-known are the human readable syntax, that is used in many published works on rules and the XML concrete syntax, which is a combination of the OWL XML (W3C, 2004) with the RuleML XML (RuleML, 2002) syntax, which is actually used for many applications.

In the human readable syntax a rule has the form (5.3):

$$antecedent \rightarrow consequent \quad (5.3)$$

where both *antecedent* and *consequent* are conjunctions of atoms written A_1, A_2, \dots, A_n . Using this syntax a rule (5.4) asserting the transitivity of the parallelism would be written:

$$hasParallel(?x,?y) \wedge hasParallel(?y,?z) \rightarrow hasParallel(?x,?z) \quad (5.4)$$

XML concrete syntax is used for different practical implementations and has an advantage, that rules and ontology axioms can be freely mixed and written in the same file. A rule in the XML concrete syntax consists of two parts. The RHS part can be found under the `<swrl:head>` element, where the conclusion of the rule is placed (the consequent in the human readable syntax). The LHS part, i.e. the body of the rule is placed under the `<swrl:body>` element (*antecedent* in the human readable syntax).

An example of the same rule asserting the transitivity of the parallelism using the XML concrete syntax is presented in Figure 5-6.



Figure 5-6: Parallelism rule (2) in the XML concrete syntax

The reasoning support of SWRL is limited, because it provides monotonic inference only and therefore negation as failure and some other functionality are not supported. Therefore SWRL rules cannot be used to modify existing information in the ontology. As a

result it is not possible to retract or remove information from the ontology with the SWRL rule.

For example, assuming following SWRL rule that can be used to increase the weight of some construction element by one (5.5):

$$Element(?e) \wedge hasWeight(?e,?weight) \wedge \quad (5.5)$$

$$swrlb:add (?newweight,?weight,1) \rightarrow hasWeight(?e,?newweight)$$

A successful invocation of this rule add the second assertion for the *hasWeight* property for the *Element* (by preserving the first property assertion), which is not semantically correct, because normally one element cannot have two different weights at the same time. Therefore this simple example shows why SWRL should not be used to modify existing information in the ontology.

Negation as failure is also not supported in the SWRL and therefore following rule (5.6), for example, is not possible:

$$Process(?p) \wedge \neg hasNext(?p,?n) \rightarrow LastProcess(?p) \quad (5.6)$$

All these facts were the main reasons, why SWRL, which at first glance may seem to be the most suitable solution for the integrations of rules and ontologies, was not chosen in this work for the practical implementation.

Jena Rules

Jena Rules are another example of the homogenous approach in the integration of ontologies and rules. Jena is an open source Java framework²¹ providing an API for working with RDF graphs and ontologies. It provides environment for RDF, RDFS, OWL and SPARQL and includes a rule-based inference engine.

Jena rules are based on RDFS and use the triple representation of RDF descriptions. A rule in Jena rules is defined by a Java object Rule with a list of body terms (premise) and a list of head terms (conclusion). Even that Jena rules and ontology axioms cannot be written in the same file (as it is possible in SWRL for example), rules are still use the vocabulary of the ontology and therefore any additional transformation between ontological knowledge base and the rule-base is not necessary.

²¹ <http://jena.apache.org> Retrieved 2013-06-14

Jena rules can be specified in a text files. The abstract syntax of Jena Rules²² is shown in Figure 5-7

```

Rule      :=  bare-rule .
           or  [ bare-rule ]
           or  [ ruleName : bare-rule ]

bare-rule :=  term, ... term -> hterm, ... hterm    // forward rule
           or  bhterm <- term, ... term             // backward rule

hterm     :=  term
           or  [ bare-rule ]

term      :=  (node, node, node)                    // triple pattern
           or  (node, node, functor)                // extended triple pattern
           or  builtin(node, ... node)              // invoke procedural primitive

bhterm    :=  (node, node, node)                    // triple pattern

functor   :=  functorName(node, ... node)           // structured literal

node      :=  uri-ref                               // e.g. http://foo.com/eg
           or  prefix:localname                     // e.g. rdf:type
           or  <uri-ref>                            // e.g. <myscheme:myuri>
           or  ?varname                             // variable
           or  'a literal'                          // a plain string literal
           or  'lex'^^typeURI                       // a typed literal, xsd:* type names supported
           or  number                               // e.g. 42 or 25.5

```

Figure 5-7: Abstract syntax of the Jena rules

Using this syntax a rule asserting the transitivity of the parallelism would be written:

@prefix ex: <http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#>.

@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

[ruleExample: (?x ex: hasParallel ?y) (?y ex: hasParallel ?z) → (?x ex: hasParallel ?z)]

²² <http://jena.apache.org/documentation/inference> Retrieved 2013-06-16

The approach with the Jena rules was used in the extension of the current work described in (Benevolenskiy, et al., 2014).

5.4.3. Hybrid approach

The main disadvantage of the hybrid approach is the need of an appropriate transformation from the ontological description into the rule-engine format, because of the fact that concepts from the ontology cannot be used directly as facts in the rule engine. A possible solution is to integrate the fact base and the ontology so that the rule engine can see and process them as one component. The proposed integration of the rule engine and the Process Instance Ontology is shown in Figure 5-8 below. The concepts from the Process Instance Ontology are transformed by the Process Configurator into Java classes with appropriate sets of properties and then with the help of the Jena API are instantiated with the corresponding individuals from the ontology. These Java classes are stored in the facts base (*Working memory*) of the rule engine. The rule engine matches the rules stored in the rule base (*Production memory*) against the facts, which are kept in the fact base.

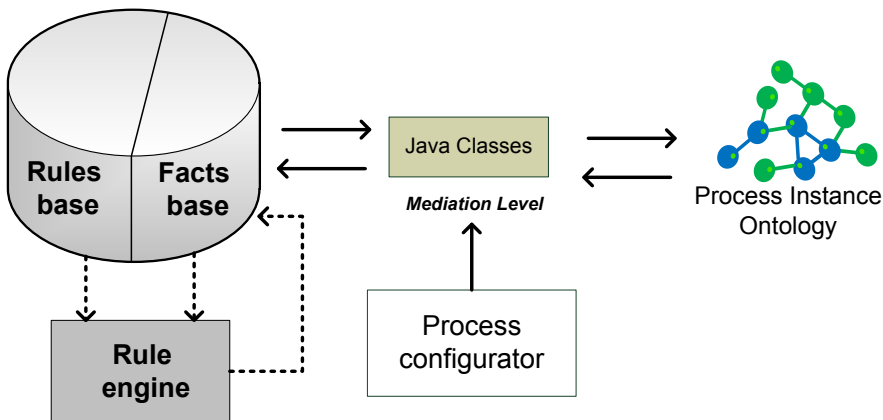


Figure 5-8: Integration of the rule engine with the Process Instance Ontology

A two-way interaction between rules and the ontology is presented in Figure 5-8. The mediation layer should support thereby the parsing and editing of the ontology, as well as the control over its instances. On the other hand, the mapping from ontology concepts in the rule engine *Working memory* also should be provided in this layer. The process configuration variants produced by the rule engine should be in turn stored in the Process Instance Ontology.

The required integration is the main reason why the implementation of the hybrid approach is certainly more difficult than a homogeneous one; on the other hand, these solutions provide more powerful inference mechanism, which can play a crucial role for some projects.

Drools

Drools is an open-sourced rule engine, which syntax supports advanced rule constructs, that was chosen for the implementation of the hybrid approach. Drools implements and extends the Rete algorithm (Forgy, 1982) and is written and tailored for the Java language (Bali, et al., 2009). The Rete is a pattern matching algorithm, which supports forward chaining and inferencing and is used for the implementation of the production rule systems.

The solution with Drools is an example of a hybrid approach in the integration of ontologies and rules, because the interaction between ontologies and rules is ensured by means of mediation code, integrated in the Process Configurator. In a complex construction project with many business processes, their features and heterogeneous resources it is crucial to represent the business logic clearly and efficiently. Such a framework is therefore a preferred solution that can successfully handle the resulting complexity.

Drools rule engine, for example, provides meaningful mechanism for the presentation of the rules. It is based on First Order Logic and therefore it also contains a negation. In some rule languages, coupled with ontologies, this is not the case. However, it may be important in a construction project to express what should be done, if certain resources are not available, or if there is no possibilities for a certain process module to be configured. Therefore the negation may be required for some specific purposes. The extra advantage of Drools is a tight coupling with Java (the Right Hand Side of the rule can be a Java expression and in Working memory the normal Java classes can be saved), what makes the implementation more flexible.

Drools rules have their own syntax which can be seen as a mixture of some logical programming language with the Java language. The basic structure a rule in Drools is similar to the one from other approaches ($LHS \rightarrow RHS$):

```
rule"name" attributes
when  LHS then RHS
end
```

In Figure 5-9 an example of a simple Drools rule is shown. This rule finds a construction element *IFCElement* with the maximum value of the coordinate *Y* in some group

IFCGroup, which has a *groupId*. This rule can be used in different more complex rules for the grouping of elements depending on their location at the construction site.

As it can be seen some Java constructs as well as a specific Drools elements are mixed in this rules. Moreover, this rule uses external parameters (like *ArrayList all* or *String groupId*) and calls some Java methods in the program from which it is executed (it adds new elements in the *ArrayList all* and prints systems messages)

```
import java.lang.Double;
global java.util.ArrayList all;
global String groupId;
/*1*/rule "Find max y"
dialect "java"
when
  $e1:IFCElement ($id:ID, y>=0)
  $g: IFCGroup(groups contains $e1.ID)
forall (($e2:IFCElement (y>$e1.y))&&($g.groups) contains $e2.ID))
then
  if ($g.ID.equals(groupId)){
    all.add($e1.ID);
    System.out.println("Element ID: "+$e1.ID);
    System.out.println("Element Y: "+$e1.y);
    retract($e1);
    System.out.println("Retracted");System.out.println("Group "+$g.groups);
  }
end
```

Figure 5-9: Drools rule finding element with maximum y value

JESS

Jess is another Java rule engine that can be used for the implementation of the hybrid approach in the integration of ontologies and rules. It uses an enhanced version of the Rete algorithm (Forgy, 1982), which has some specific optimizations related to the performance as well as the support of a backward chaining, to match rules against the Working memory. Jess as well as Drool also supports two main classes of rules: forward-chaining and backward-chaining rules. It has its own syntax to define rules and can be embedded in the Java applications.

Jess can be seen as a pure hybrid approach, however its plugin – JessTab provides a tight integration between an ontology and the rule engine and therefore in some cases it can be considered as an example of the homogenous approach too. JessTab is a bridge between Jess rule-engine and an ontology that allows mapping of the ontology

knowledge base to Jess facts so that Jess rule patterns can match on ontology individuals.

An example of a simple Jess rule printing a construction element with corresponding id that belongs to some construction process is shown in Figure 5-10. Even that Jess rule engine is also written in Java, it cannot provide the same flexibility and functionality as Drools rule engine do (for example the realization of the external methods call is more complicated) and therefore Drools was chosen as a preferred solution.

```
(defrule print_process_and_object
(object (is-a Process)
(OBJECT ?p)
(id ?id))
(object (is-a ifcWall|ifcColumn|ifcSlab)
(is-a-name ?type)
(belongsTo ?p)
(id ?oid))
=>
(printout t "Process with id:" ?id " uses construction object of type " ?type " with an id "
?oidcrlf))
```

Figure 5-10: Jess rule that finds a specific object of some construction process

5.4.4. Types of rules

In the current work three main types of rules are considered:

- *Sequence check rules*: check whether the proposed sequence is correct;
- *Configuration rules*: generate one sequence of activities;
- *Strategic rules*: generate more complex sequences of activities according to some configuration strategy.

Sequence check rules are the simplest type of rules in this work that can be used to define the order of the executions of the subprocesses in a process module. This kind of rules usually cannot perform any configuration or calculation tasks, but just check whether the proposed sequence of the activities is correct and inform the user about the result. These rules can be used to check new process patterns.

One example of a sequence check rule that can check the sequence of the construction operations is presented below. The operation *Concrete* has to be performed after the operation *Form*, if they both operate on the same construction element.

Example (Formal syntax):

- $\text{hasNext}(x, y) \rightarrow \text{hasFollowing}(x, y)$
- $\text{Concrete}(x) \wedge \text{hasObject}(x, z) \wedge$
- $\text{Form}(y) \wedge \text{hasObject}(y, z) \wedge \text{hasFollowing}(x, y)$
- \rightarrow
- $\text{Print}(\text{"Error process:"} + y + \text{" must be performed before process:"} + x)$

Configuration rules are the mostly used type of rules in this work. They are used to generate and configure the sequence of construction processes depending on various configuration parameters. These rules also can be used to combine different processes in groups according to their priority, resources availability or location of the corresponding construction objects.

An example of a configuration rule is the process of constructing storeys according to their elevation. That means that storeys with the lower elevation are constructed before storeys with the higher one.

Strategic rules are the most complex rules that are used in this work to generate sequence of activities as well as different variations of them based on some configuration strategy. More detailed description of configuration strategies as well as some examples of them are presented further in a separate section (Section 5.4.5).

Hierarchical approach

In the current work a hierarchical approach is used in the implementation of the rules (Figure 5-11). This means that for every process-level the relevant rules/strategies have to be developed (building, storey, group of elements, elements, items). Such an approach allows the construction processes to be easily top-down configured, as well as to be cancelled at any stage of configuration. The rule mechanism is very flexible, so that combining rules from the same level is possible. Therefore the user can simultaneously see the impact of a few configuration rule sets. For example, taking into account technological constraints like *"Build walls and columns before slab"* and *"Build slab before stairs"*, reduces the number of configuration variants essentially. On the next level such rules as *"Build all columns by Travelling Salesman Algorithm"* would be of particular interest. Finally on the last level the rules, enabling the parallelization of the process, would be implemented. Which processes can be finally executed in parallel depends on the existing resources (equipment, materials, construction workers, etc.) and other influence factors. For example, by setting the number of the available formwork elements it is possible to define how many forming works could run in parallel.

In such a hierarchy of rule sets, it is complicated to evaluate them all at the same time at the end. However, in a complex building system, in which many components and factors play a role, it is crucial to divide this complexity in several levels.

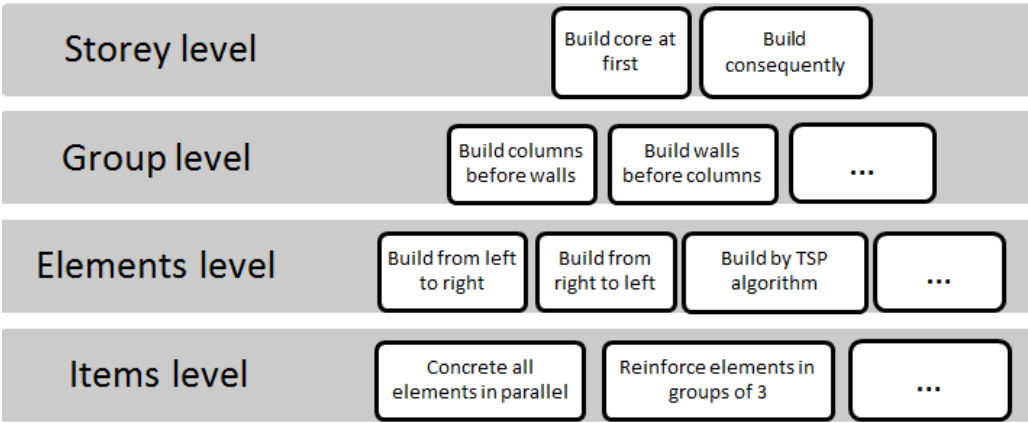


Figure 5-11: A hierarchy of the rules

Rules overview

In Table 7 an overview of some implemented rules with their description and application level is shown. The full list is presented in the appendix A. In the last column of the table authors of rules are presented (for Benevolenskiy (Benevolenskiy A. , Roos, Scherer, & Katranuschkov, 2014) and for Roos (Roos K. , Benevolenskiy, Katranuschkov, & Scherer, 2014))

Most of the presented rules were implemented for a general purpose rule engine Drools. For the implementation of the Traveling Salesman Problem algorithm the constraint satisfaction solver OptaPlanner²³ was used.

Table 7: Overview of some rules

| Name | Level | Description | Implementation |
|--------------------------|--------|--|------------------------|
| Build consequently | Storey | Construct storeys of the building consequently according to their elevation | Roos/ Benevolenskiy |
| Build core at first | Storey | Construct at first core elements of several storeys and then the rest elements (s. Core and Rest configuration strategy) | Benevolenskiy/ Roos |
| Group core elements | Group | Group all core elements of the building in respective groups (s. Core and Rest configuration strategy) | Benevolenskiy |
| Group rest elements | Group | Group all rest elements of the building in respective groups (s. Core and Rest configuration strategy) | Benevolenskiy |
| Group by distance matrix | Group | Group elements according to the distance from each other (s. Group and sequencing configuration strategy) | Benevolenskiy |

²³ www.optaplanner.org Retrieved 2012-03-25

| | | | |
|--|----------------|---|--------------------------------|
| Construct walls/columns/ group at first | <i>Group</i> | Construct elements of the selected group at first, other groups will be constructed afterwards (combinatory rule) | <i>Roos</i> |
| Build by Horizontal Matrix | <i>Element</i> | Build elements in the order according to their position in the storey (coordinate X) | <i>Roos/ Benevolenskiy</i> |
| Build by TSP algorithm | <i>Element</i> | Order the elements/processes in some group according to the Traveling Salesman Problem algorithm | <i>Roos</i> |
| Concrete/Task in parallel | <i>Item</i> | Execute concreting/other works in parallel, other tasks will be executed sequentially | <i>Roos</i> |

5.4.5. Configuration strategies

Before discussing about different configuration strategies used in this work it is necessary to define what do configuration and configuration strategy in the context of this work mean. For these purposes an intuitive definition of the notion configuration that fits a large number of design tasks from different areas provided in (Mittal & Frayman, 1989) was chosen:

Configuration is a special type of design activity, with the key feature that the artifact being designed is assembled from a set of pre-defined components that can only be connected together in certain ways.

In our case the artifact that is being designed is a sequence of construction processes composed from a set of pre-defined components, which are process patterns.

Three main existing paradigms for the configuration systems were proposed in (Sabin & Weigel, 1998):

- *Rule-based reasoning approaches* (also known as expert systems) use production rules for representing configuration strategies and deriving solutions. Numerous configuration systems in the artificial intelligence and related areas were developed using this approach. The configuration solution proposed in this work

is also an example of the implementation of the rule-based reasoning approach for the configuration.

- *Model-based reasoning approaches* assume the existence of a system's model consisting of decomposable entities with interactions between them. There are several model-based approaches to configuration including: logic-based approaches, resource-based approaches and constraint-based approaches.
- *Case-based reasoning approaches* store earlier configuration solutions in the form of so called cases. The new configuration problem is solved based on the solutions of similar past problems described in the cases. Each new solved configuration problem is then added to the knowledge base of the system as a new case.

Under a term *strategy* a plan of actions intended to achieve one or more goals is understood. In this work a lot attention is paid to a so called configuration strategy.

A *configuration strategy* is a plan of actions, with a specific set of construction objectives, preferences or conditions designed to achieve a particular configuration goal. In this work the realization of strategy is a set of rules used for a special configuration goal. The goal is usually to finish the construction project with the minimal time, costs or resources used.

Some of the strategies are used to configure processes in a way that the total execution time is minimized. This can be achieved when several rules to parallel different processes are applied. Other strategies can be used to configure processes with limited number of resources. In the following section two simple examples of some of configuration strategies implemented in this work are presented. These strategies represent configuration strategies only for some specific level of details. The overall configuration strategy consists of several strategies defined for each level of details.

Core and Rest configuration

The so called "*Core and Rest*" configuration strategy is based on the assumption, that core elements of the building should be constructed at first and the remaining elements can be constructed after them. This can be the case, when the construction of the core elements can be done faster by using some special equipment or resources. For example the use of the special large formwork systems allows constructing walls in one building through few storeys in one procedure, which can result in the minimization of construction time and costs. To illustrate this strategy a construction of the high-rise office building is considered in the example shown in Figure 5-13.

Construction elements of the building are sorted into two groups. The first group is called *Core elements*. It includes all structural walls that are constructed in this building by using a special large formwork system. All remaining elements belong to the second

group called *Rest elements*. In this configuration strategy it is possible to set a parameter defining a so called shift (a difference in the construction of core and remaining elements). In the example presented in Figure 5-13 this parameter is equal 3, which means that the remaining elements on the first storey can be constructed only after constructing of the core elements on the first three storeys.

This configuration strategy is defined for the storey and group process-levels (s. Hierarchical approach) and consists of few rules. First rule is used to sort elements on all storeys into two groups depending of their type. Other rules are used to control the construction of the core and rest groups with a required parameter. The simplified flowchart of this configuration is shown in Figure 5-13 to the right.

Grouping and sequencing configuration

"Group and sequencing" is a more complex configuration strategy that is used to group construction elements of the same type in one storey into two groups and to generate optimal execution sequences of processes in these groups. This configuration operates on the group and elements process levels and can be also used together with the *Core Rest* configuration presented above. In Figure 5-14 an example of this strategy for the configuration of the processes constructing columns on some storey is presented.

In the first step the distance matrix between all columns on the storey is calculated and two columns with the maximum distance between each other are chosen. After that all columns on the storey are sorted into two groups in such a way that the distance between each column and the chosen left or right column is minimal. That results in two sets of the elements grouped near the chosen most right and most left columns. In the last step the order of the execution of the processes is defined (shown in Figure 5-14 with numbers: first 1 than 2 and so on) so that at first columns on the on sides are built and then columns in the center.

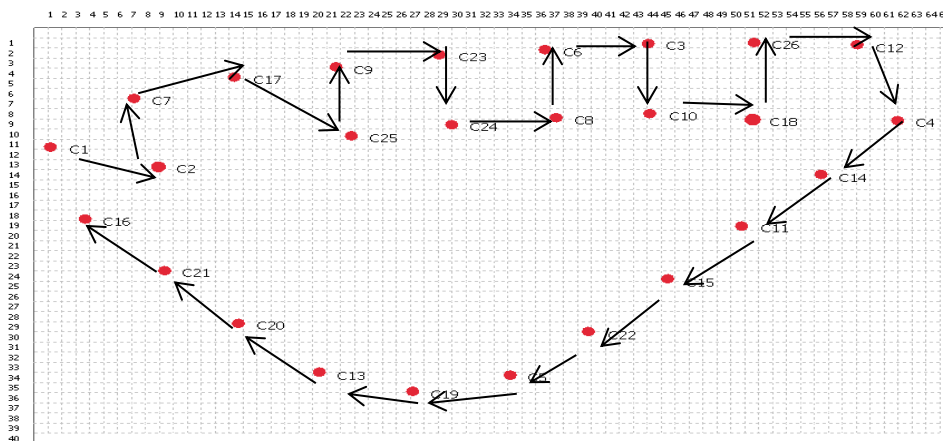


Figure 5-12: Founded execution sequence for columns

This configuration strategy also consists of many rules including specific subrules used for grouping and sorting of elements as well as subrules used for generating executions sequences. For example for the generation of the execution sequences an algorithm based on the travelling salesman problem (TSP) can be used. It allows generating the shortest possible sequence of processes, such that each object is visited exactly once and after that the algorithm returns to its starting point. In Figure 5-12 a founded solution with the TSP algorithm solution for the optimal sequence of column constructing processes at one floor is shown.

Another example is the usage of the rules enabling the parallelization of the processes. In that case the elements can be divided in several groups in which they can be executed in parallel. The number of processes that can be executed in parallel depends normally on the availability of some particular resource and can be set as a parameter for the configuration.

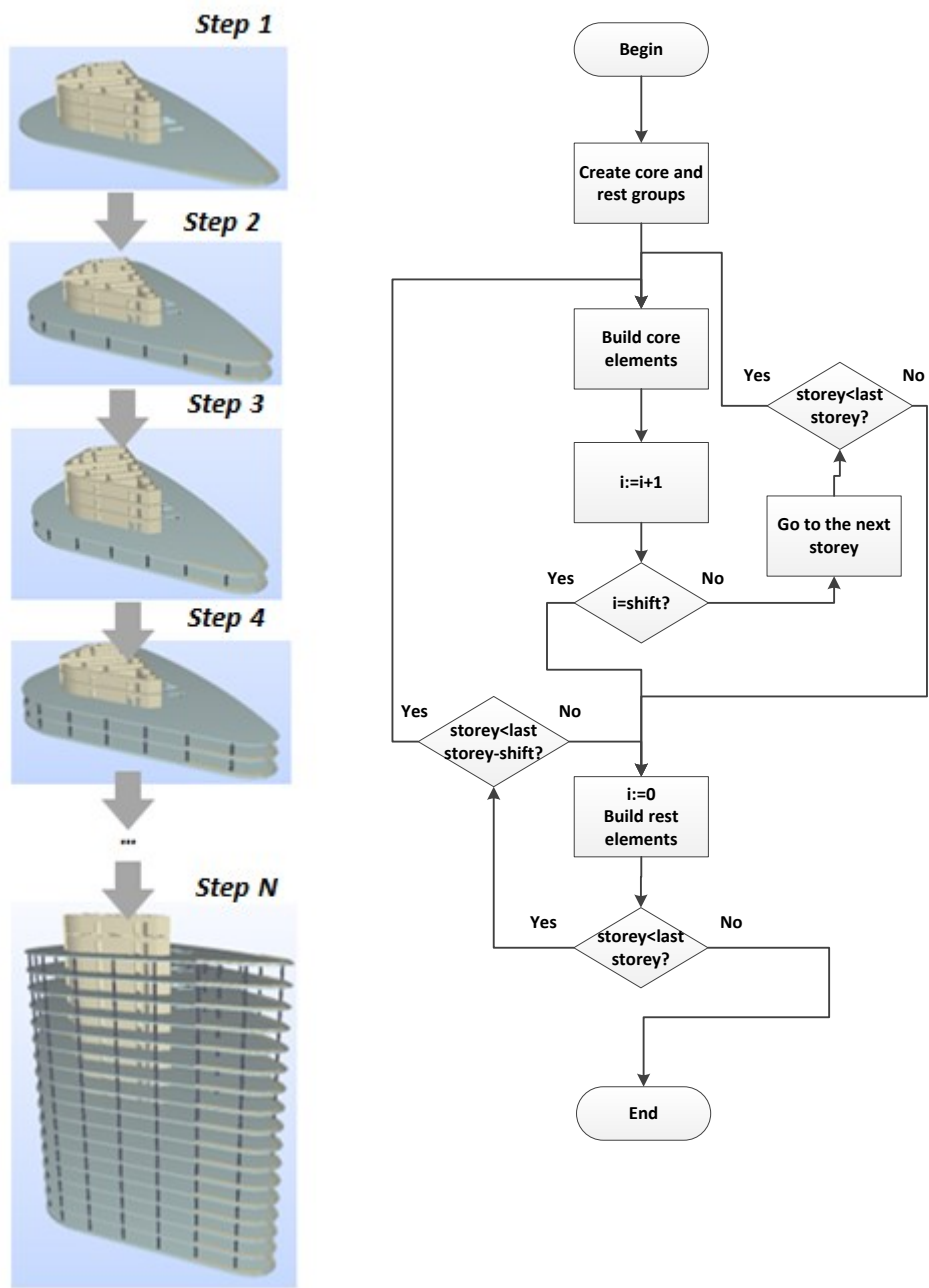


Figure 5-13: Core/rest configuration with its flowchart

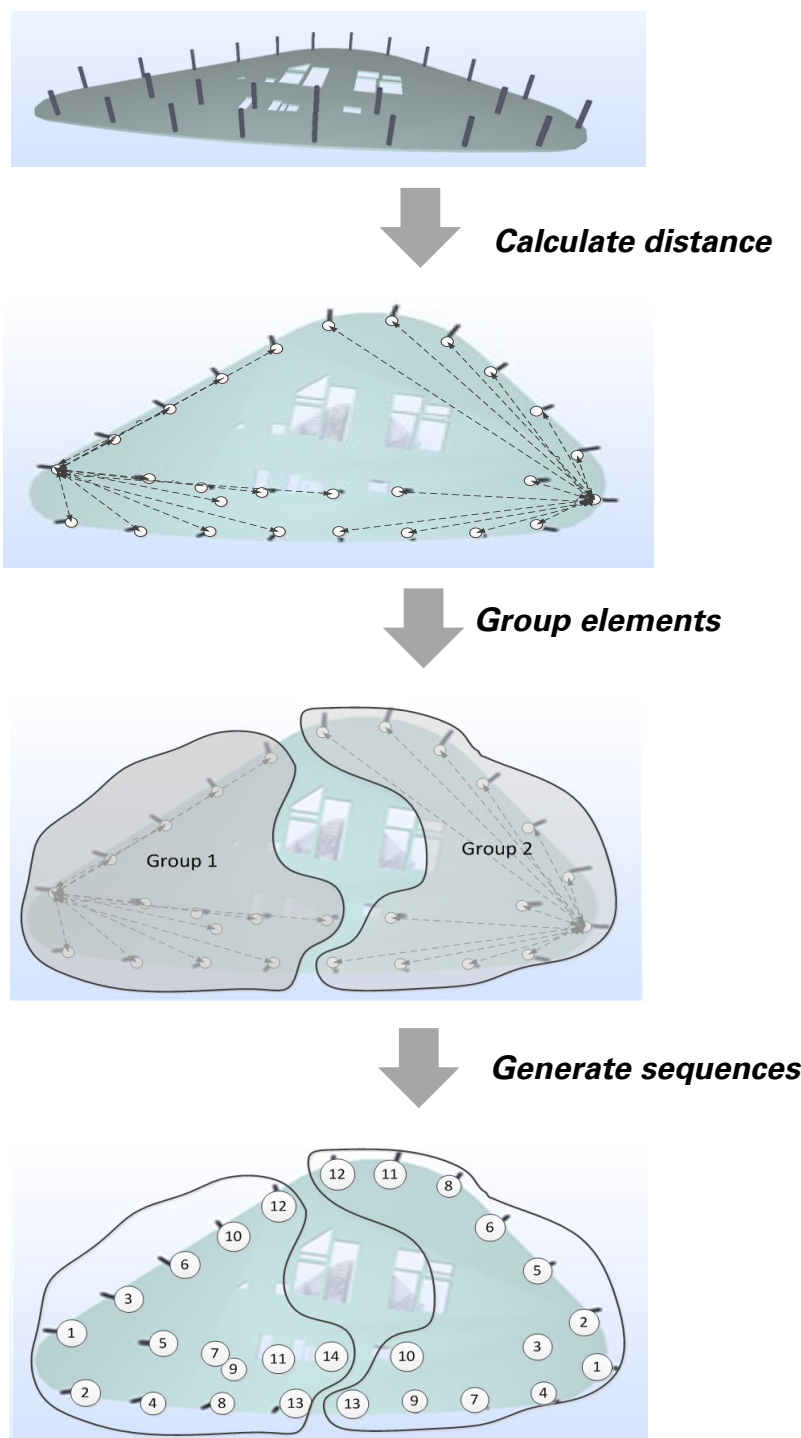


Figure 5-14: Configuration strategy for columns on one storey

6. Implementation

This chapter provides a description of the overall system architecture and presents the implementation of the software prototype.

The system architecture with all its components is demonstrated and described in detail in the first section. There the relations between ontologies, process pattern editor, rules-engine and Process Configurator are presented. The second section is dedicated to the Process Configurator, which is a software prototype supporting the generation of process schedules for construction projects with the help of reusable process patterns and configuration rules. Process Configurator realizes the interaction between the ontologies, described in the chapter 4 and implements the configuration components of the system, presented in the chapter 5.

The mapping mechanism between an ontology-based format and BPMN format of the construction processes, which is used in different steps in the Process Configurator, is considered in the last section.

6.1. System architecture and its components

After describing ontologies and the configuration method it is necessary to demonstrate the whole structure of the proposed system with all its components and relations between them. Therefore in this section all major elements of the system, that were described in the previous chapters as separate components will be presented together in one structure and the interaction between these components will be demonstrated. The suggested overall system architecture is shown in Figure 6-1 below.

The main idea of the approach proposed in this work is the development of a formal high-level model for the construction processes realized by means of two ontologies and their integration with the general-purpose rule engine used for the configuration of these processes.

These two ontologies, i.e. a *Process Pattern Ontology* and a *Process Instance Ontology*, are separate but inter-related in such a way, that the process patterns from the first ontology can be used in many different construction processes represented by specific instantiations of the second ontology. The detailed description of the both ontologies and their main components is provided in chapter 4 of this thesis.

As an input BIM model a standard IFC-based model data is used and imported into the ontology(1).

The ontological models are queried through SPARQL, an RDF query language (s. chapter 3). Pattern retrieval is also realized with the help of SPARQL, enabling the dynamic binding of search attributes/concepts (2). The structure of the pattern and the retrieval process are described in chapter 4 und chapter 5.

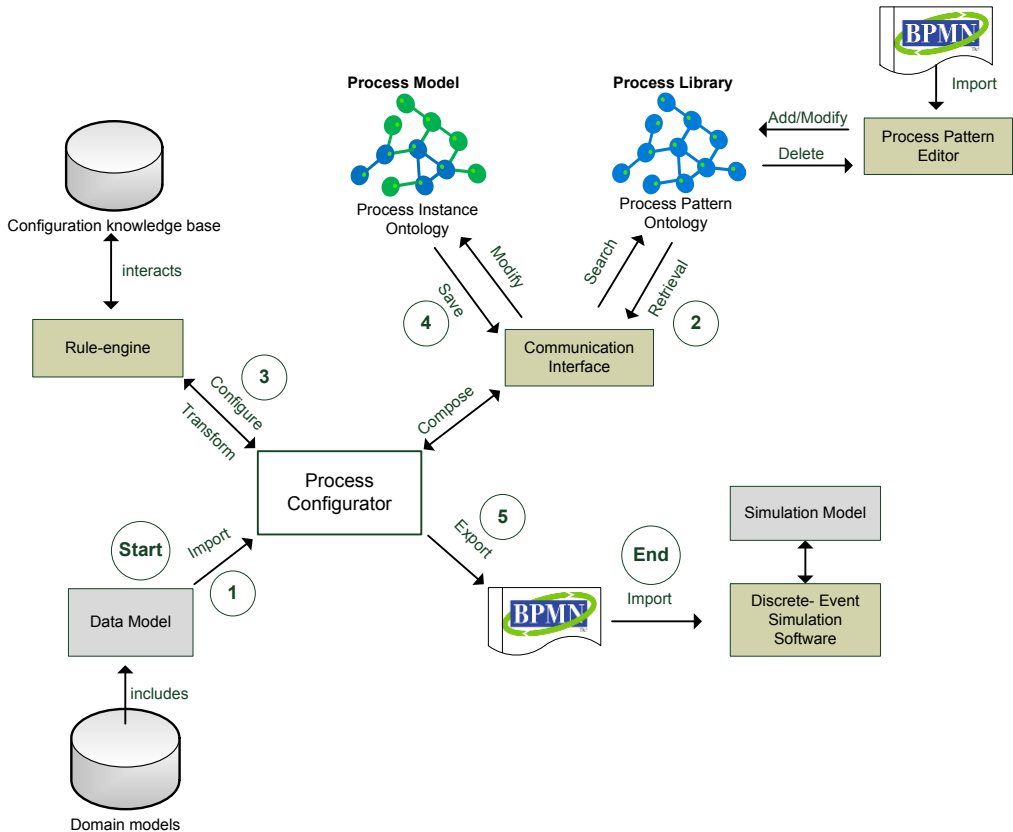


Figure 6-1: Principal system architecture

The Process Pattern Ontology manages and stores the hierarchy of the generic process patterns. These patterns comprise configurable sequences of subprocesses as well as mathematical expressions (parameters) to calculate process duration time and estimate their production costs.

The Process Instance Ontology is populated with the instantiated construction processes (process modules) and is unique for each construction project. Both ontologies are described by a common set of concepts in order to support their interaction. To work with the ontologies the Jena framework (Open Source Semantic Web Framework for Java) is used. It provides a programmatic environment for RDF, RDFS and OWL.

The Process Pattern Editor is used to add new and modify existing process patterns in the Process Patterns Ontology. New process patterns that are imported in the Process Editor are serialized in the BPMN format. The mapping procedure is described in the following section in detail.

During the process configuration the content of the Process Pattern Ontology remains unchanged, because the ontology is only used for the search and retrieval of the process patterns. However, in the meantime the content of the Process Instance Ontology is changing constantly, because process modules are instantiated or modified during the configuration.

In order to configure the instantiated modules from the Process Instance Ontology different configuration rules are used (3). For the rule execution in this work an external powerful Java-based rule engine is used. The rules are stored separately from the rule engine in text rule-files. A single rule-file can contain multiple rules as well as some resource declarations like imports and attributes that are used in rules. Rules cannot be applied directly to the ontological description of the process in OWL. Therefore, the Process Configurator performs the required process transformation from OWL into the appropriate rule-engine format (4).

Finally, after the configuration phase is done within the Process Configurator, the configured construction process can be simulated by discrete-event simulation software, where the work schedules for the corresponding construction tasks are generated (5).

6.2. Process configurator

The proposed configuration system, i.e. the interaction between the ontologies and other components is realized by a Java-based *Process Configurator*, which has the purpose to support top-down process planning while continuously considering respective building model data, as well as resources, equipment and overall strategic parameters. During the configuration process the user works with the graphical user interface (GUI) of the Process Configurator (shown on Figure 6-2).

Ontology-based configuration of the construction process in the Process Configurator consists of the following three principal steps:

- Construction of general models;
- Adapting the process model to the specific project context with the help of a knowledge base of process patterns (retrieval & adaptation step);
- Configuring the process model with the help of rules (configuration step).

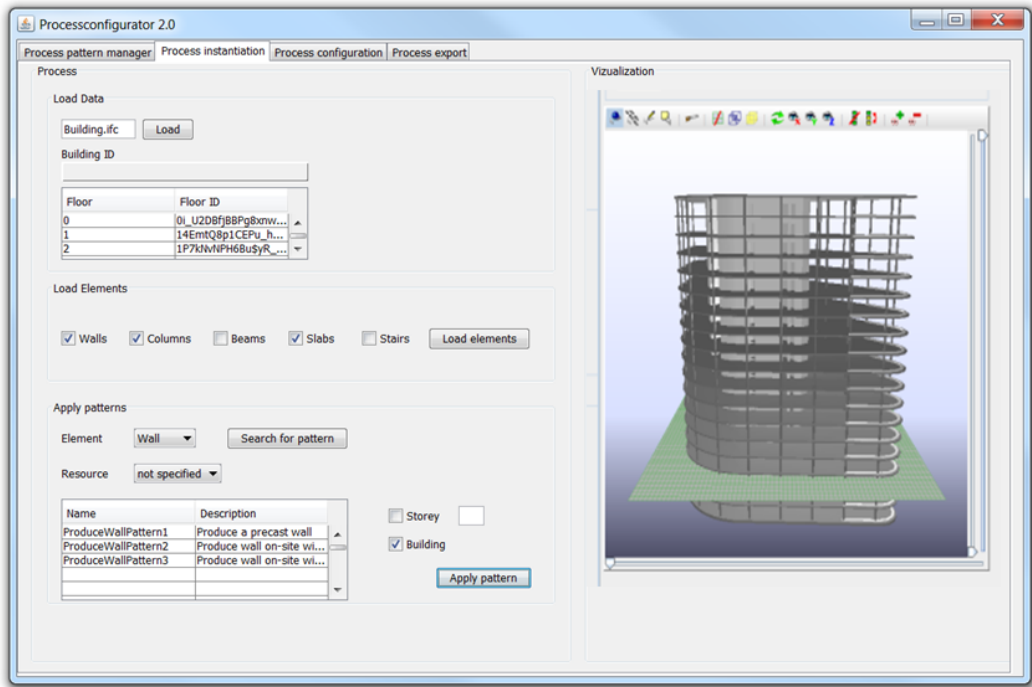


Figure 6-2: GUI of the Process Configurator and used building model

The Process Configurator consists of the few panels, which are used for different functionality:

- Process pattern manager;
- Process instantiation;
- Process configuration;
- Process export.

These panels are provided with individual GUIs, which are shown and described in the following.

Process pattern manager

The Process pattern manager provides a graphical user interface to work with the process patterns. It can show all process patterns with their descriptions and related objects saved in the Process Pattern Ontology. Moreover, it provides functionalities for importing new process patterns into the Process Pattern Ontology or exporting existing process patterns from the ontology into a single xml file. The process pattern can be defined in any software tool supporting BPMN 2.0 XML serialization and then imported in the Process Pattern Ontology. The process pattern manager with the table of the available patterns and the import/export interface is shown in Figure 6-3.

An important issue by the importing of new patterns is the ontology alignment or ontology matching problem. It has its goal in comparing different new subprocesses and determining correspondence between them and available processes in the ontology. In the current implementation we just used an implicit words comparison. For the further extensions more comprehensive methods and techniques can be used, as nowadays numerous researches are conducted in this area.

Scalability

The amount of instances in the ontology is closely related to the scalability problem. The scalability of the ontology depends normally on the technical implementation of the triple store, which is used for storing and querying RDF data. Depending on the architecture of the implementation three categories of triple stores can be defined: in-memory, non-memory native and non-memory non-native. In this implementation an in-memory-based model was sufficient, because of the relatively small amount of instances in the ontology, however it is necessary to note that for storing extremely large volumes of data a non-memory solutions should be chosen.

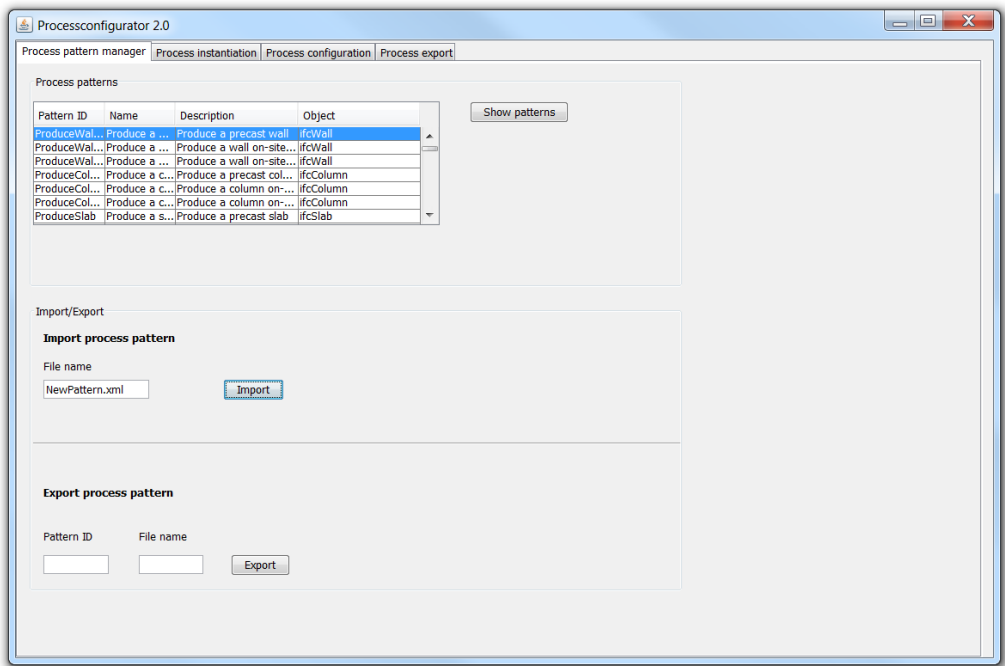


Figure 6-3: Process pattern manager

Process instantiation

Process instantiation panel provides a user interface for importing a source data model and constructing a new general model that will be used later for the configuration. At first the source data model is loaded. By selecting specific construction element types in the GUI (*Load Elements* section) the user has a possibility to choose how detailed the constructed general model should be and which elements it should contain. By default, the model is loaded without any construction elements (only building with all its floors is loaded), but required elements can be loaded by selecting respective checkboxes in the panel (*Walls, Columns, Beams, Slabs, Stairs*).

Furthermore, process pattern search and adaptation, described in the chapter 5, take place in this step. The user has a possibility to search for the process patterns stored in the Process Pattern Ontology and filter them according to resources used in these patterns. After a required pattern is found, it can be applied to all elements of the respective type in some specific storey or in the whole building. Therefore various patterns can be used to construct building elements in different floors.

On the right side of the Process instantiation panel a 3D-vizualisation of the source data model is presented.

The Process instantiation panel is shown in Figure 6-4.

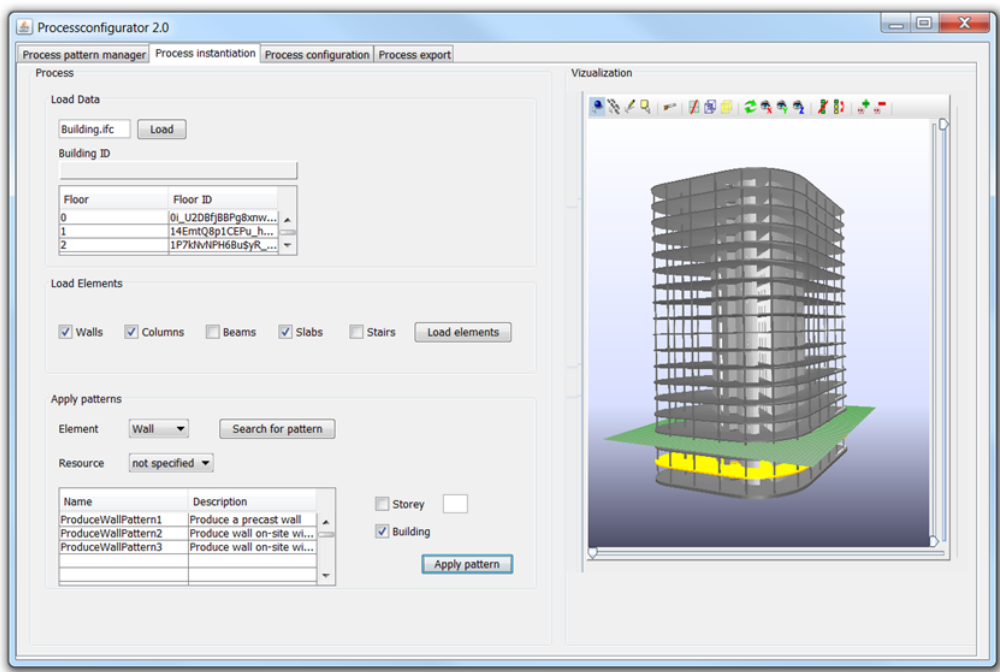


Figure 6-4: Process instantiation panel

Process configuration

Process configuration panel provides different options for configuration of construction processes. After the instantiation and adaptation steps the complete process model is saved in the Process Instance Ontology. This model is already configured by default and can be exported further for the simulation.

However, in order to get more optimal solution the user could apply different configuration strategies and reconfigure the initial process model. The theoretical basis and the detailed description of the configuration process, configuration rules and strategies are presented in the chapter 5.

In the Figure 6-5 a part of the Process configuration panel is shown. The user has here the possibility to apply following configuration steps:

- *Connect floors consequently* (Processes for construction floors in the building will be ordered and executed according to the elevation of the floor. That means that the construction process will be performed bottom-up and at first the floor with the lowest elevation will be constructed)
- *Apply grouping mechanism* (Production processes for different construction elements on the floor are grouped according to the type of the element and can be executed in the predefined order)
- *Apply Matrix method* (Production processes within groups are ordered according to their positions on the floor. As an example the implemented Horizontal Matrix method orders elements according to their horizontal coordinate)

The part of the process configuration panel presented in Figure 6-6 provides functionalities to apply *Core and Rest* configuration strategy described in the section 5.3.2. The user has a possibility to set a shift parameter (parameter is equal 2 in this example) and also apply the Matrix method, described above, for the configuration of the elements within the Core and Rest groups. At first processes at the storey level will be grouped according to the used shift parameter (*Connect Kern-Rest* section in Figure 6-6). Then further configuration is performed at the floor level (Elements section in Figure 6-6), where some specific group can be chosen (in this example one group at the fourth floor with ID *Produce_Restid2olkDaQUT9cx2jdD5ua_GJ* was selected and the *Horizontal Matrix* method was applied). All elements of the selected group can be also visualized at the 3D-building model presented at the right sight of the panel.

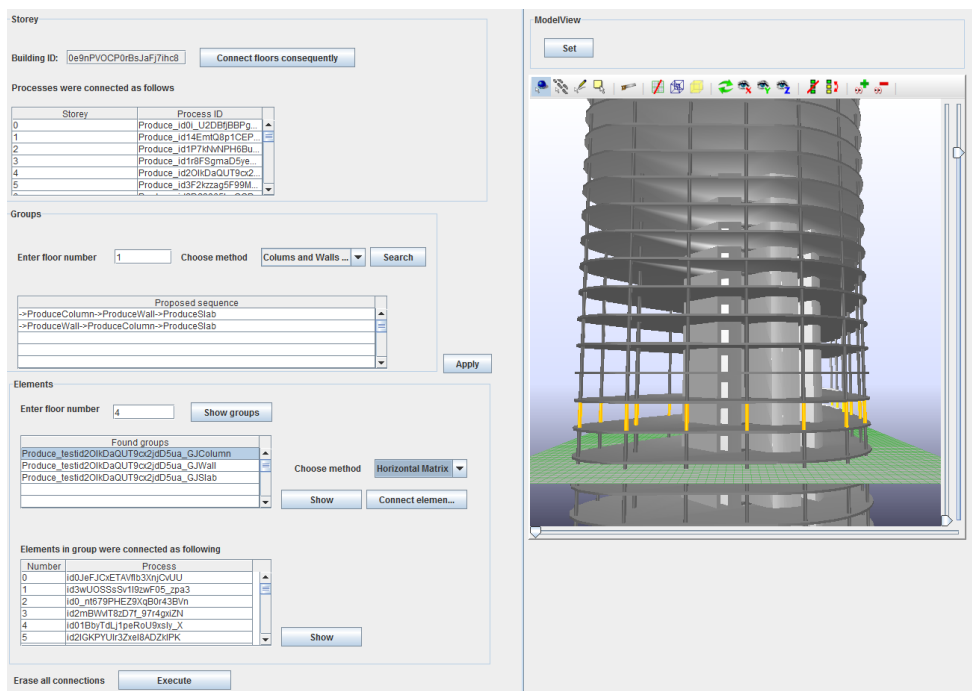


Figure 6-5: Process configuration (1)

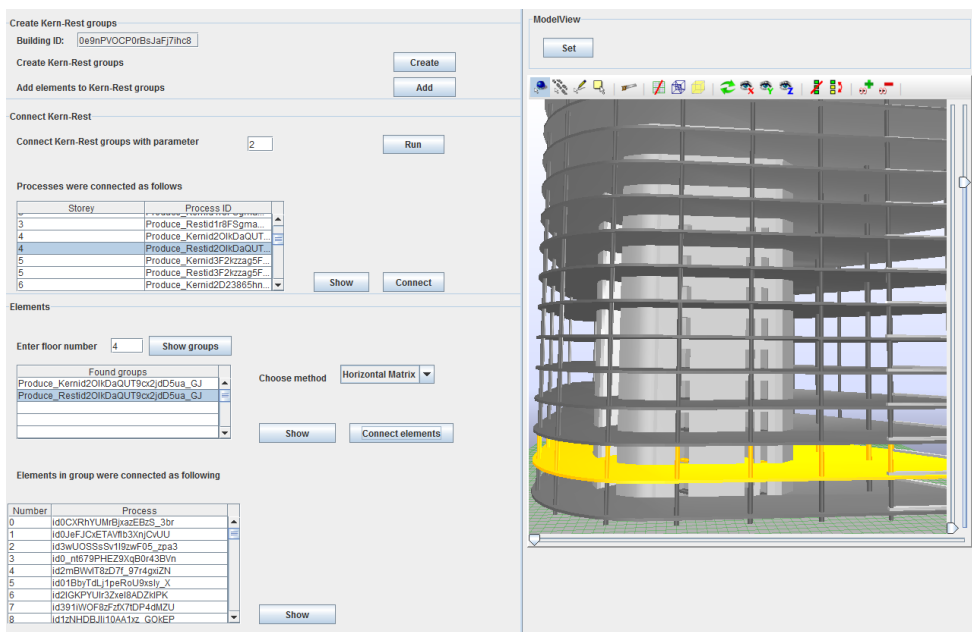


Figure 6-6: Process configuration (2)

Process export

Process export panel, presented in Figure 6-7, is used to export configured processes from the Process Instance Ontology into an XML serialized file in BPMN format. The user sets a process ID and a file name and then the required process from the ontology will be exported, transformed in the BPMN format and saved in a selected file, which can be used as an input for the discrete-event simulation software or can be visualized in any available BPMN-Viewers.

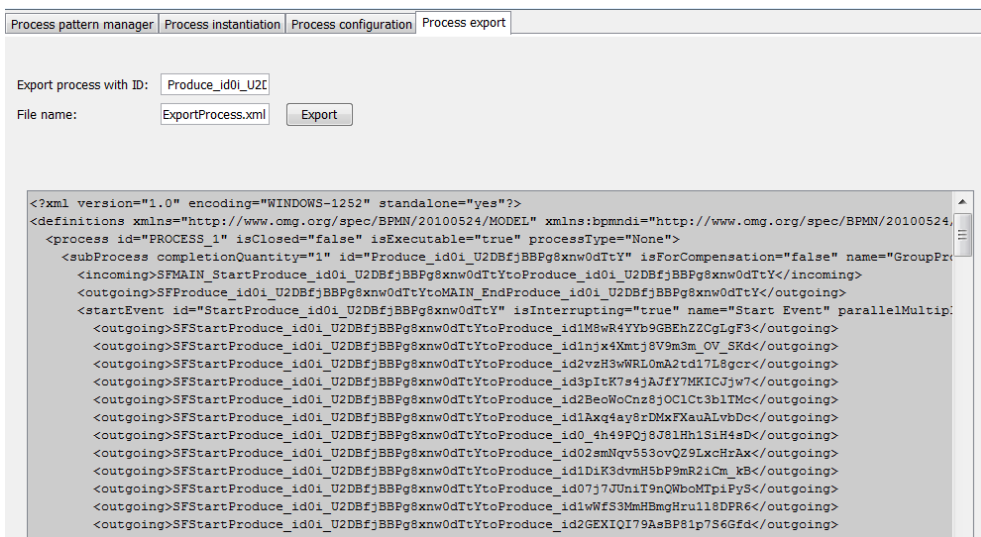


Figure 6-7: Process export panel

6.2.1. BPMN mapping

BPMN is nowadays a commonly used notation for specifying business processes. It provides not only a graphical notation depicting steps in a business process, but also an XML serialization containing the model semantics as well as the diagram-interchange information.

In this work BPMN is used in few different steps of the proposed approach. As it can be seen from the principal system architecture shown in Figure 6-1, BPMN XML serialization is used in the initial step, when the process patterns in the BPMN format are imported in the Process Configurator as well as in the last step, when the configured process modules are exported from the Process Configurator in the BPMN format.

The use of the BPMN format for these cases has a lot of benefits because users can define new process patterns in their own software tools supporting BPMN XML 2.0 and then import them into the Process Configurator. Furthermore the configured process modules exported from Process Configurator can be open and visualized in any software supporting BPMN XML serialization. Therefore even having a complex internal form for storing and processing processes in the ontologies in the Process Configurator the interaction with other systems can be easily realized through the support of the BPMN import and export functionality. To realize this functionality two mapping mechanisms were developed and implemented: mapping of the process patterns from the BPMN serialization into the Process Pattern Ontology (Import mapping) and mapping of the configured process modules from the Process Instance Ontology into the BPMN serialization (Export mapping).

Import mapping

Import mapping mechanism is used to import and transform process patterns saved in the BPMN XML serialization into the Process Pattern Ontology. The mapping process consists of several methods and cycles and is implemented in the Java programming language by using special APIs to parse XML Data and work with ontologies. One of the main features of this mapping mechanism is the constant need to check whether some classes for objects or resources already exist in the ontology and if not then create them. Moreover, the implementation of the mapping mechanism of Sequence Flows into hasNext relationships was not trivial (realized not as a 1 to 1 mapping, but as a function mapping of a set of arguments to one new relation), because of very different representation of these components.

The mapping begins with the check if the process class with the same name as importing class already exists in the Process Pattern Ontology. If this is the case, then a new instance of this class will be created, otherwise a new class with respective instance will be added. Similar procedures are used for importing process object, tasks and resources. The procedures for tasks and resources are performed in a cycle until all elements are added to the ontology. Moreover, by adding a task an addition class check, similar to the one done before is carried out. A simplified flowchart of the Import mapping is shown in Figure 6-8 to the left.

Export mapping

Export mapping mechanism is used to export process modules after their configuration from the Process Instance into BPMN XML format. This mapping process is also implemented in the Java programming language by using the same APIs as in the previous

mapping. The mapping uses few recursive methods, because the configured process modules are usually quite complex and consists of many different levels of details.

The challenge in the realization of the export mapping was that the process modules can have many different subprocesses levels and their amount and deepness is not known in advance. Therefore this method uses few recursive procedures, for example for defining tasks and subtasks. If we, for example, compare a process module with a graph in this case, we can say that the mapping is finished when all elements of this graph were visited and exported. A simplified flowchart of the Export mapping is shown in Figure 6-8 to the right.

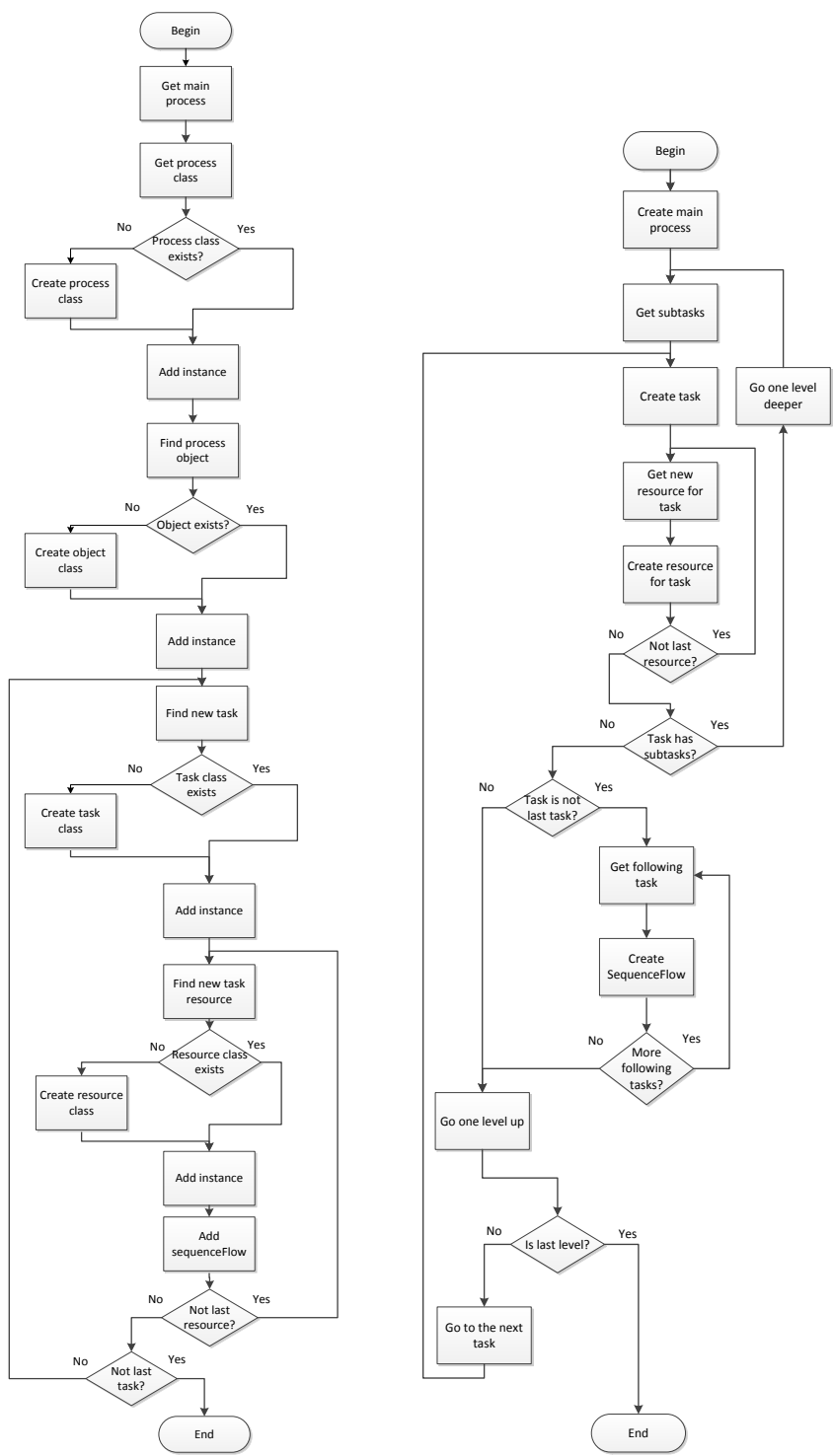


Figure 6-8: Simplified flowcharts of the import (l) and export(r) mapping mechanisms

7. Case study

In this chapter two process configuration scenarios are described. Both scenarios are based on existing project data of real-world construction projects and have their goal to show the application of the approach proposed in this research work.

In the first scenario a model of a high-rise office building is used. The building consists of eighteen storeys with different storey elements on each of them. A model of a five-storey school building is used in the second scenario.

After the presentation of the related IFC building models, some major configuration steps will be described. This includes the choice of the required process patterns and the application of the configuration strategies. Finally, the configured process sequences for both projects are shown and evaluated.

7.1. Case 1

Two models are used to demonstrate the advantages of the approach proposed in this work. In the first case for the primary validation of the system a model of a high-rise office building is used. The building consists of 18 storeys with different storey elements. The model is based on existing project data of a real-world construction project and includes information about all storeys in the building with some of their attributes and contained storey elements. In the current research the subdomain structural concrete works was considered and therefore only main storey elements such as walls, columns, slabs and stairs were taken into account. The 3-D visualization of the building is shown in Figure 7-1 .

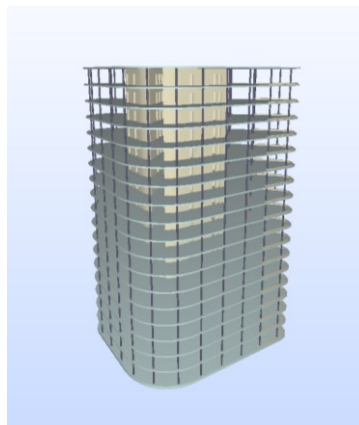


Figure 7-1: High-rise office building

The entire configuration process is performed hierarchically and consists of the three steps (shown in Figure 7-2). The process starts from the whole building and then details the construction process consequently until the building element level is reached.

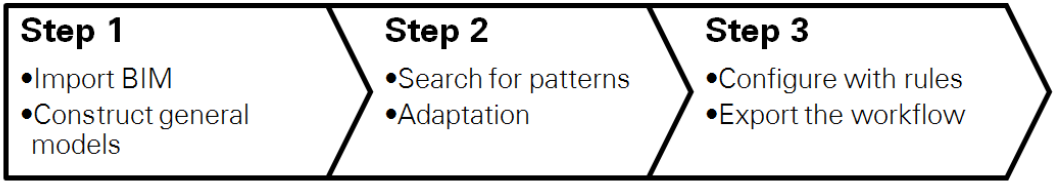


Figure 7-2: Configuration process

In the first step the required information is obtained from the input BIM (Building Information Model) and the general object and process models are constructed. Both models are saved in the Process Instance Ontology and contain only taxonomic information about the building. The general object model in the first case represents the structure of the high-rise office building and contains more than 2000 different elements and relations between them. The corresponding general process model is generated automatically in the first step and is used as a basis for the further enrichments taking place in the following steps. At this point process model includes only general process hierarchy including the top process (production of the building) and subprocesses (processes for each storey) without any further detailing. Complete process information is specified in the next steps with process patterns. For the input BIM model a standard IFC-based data model is used. The taxonomic relations between the elements are modelled with corresponding ontology relations (*contains* for the object model and *hasSubTask* for the process model). An automatically generated visualization of the general object and process model with their classes (yellow circles) and instances (purple diamonds) is shown in Figure 7-3.

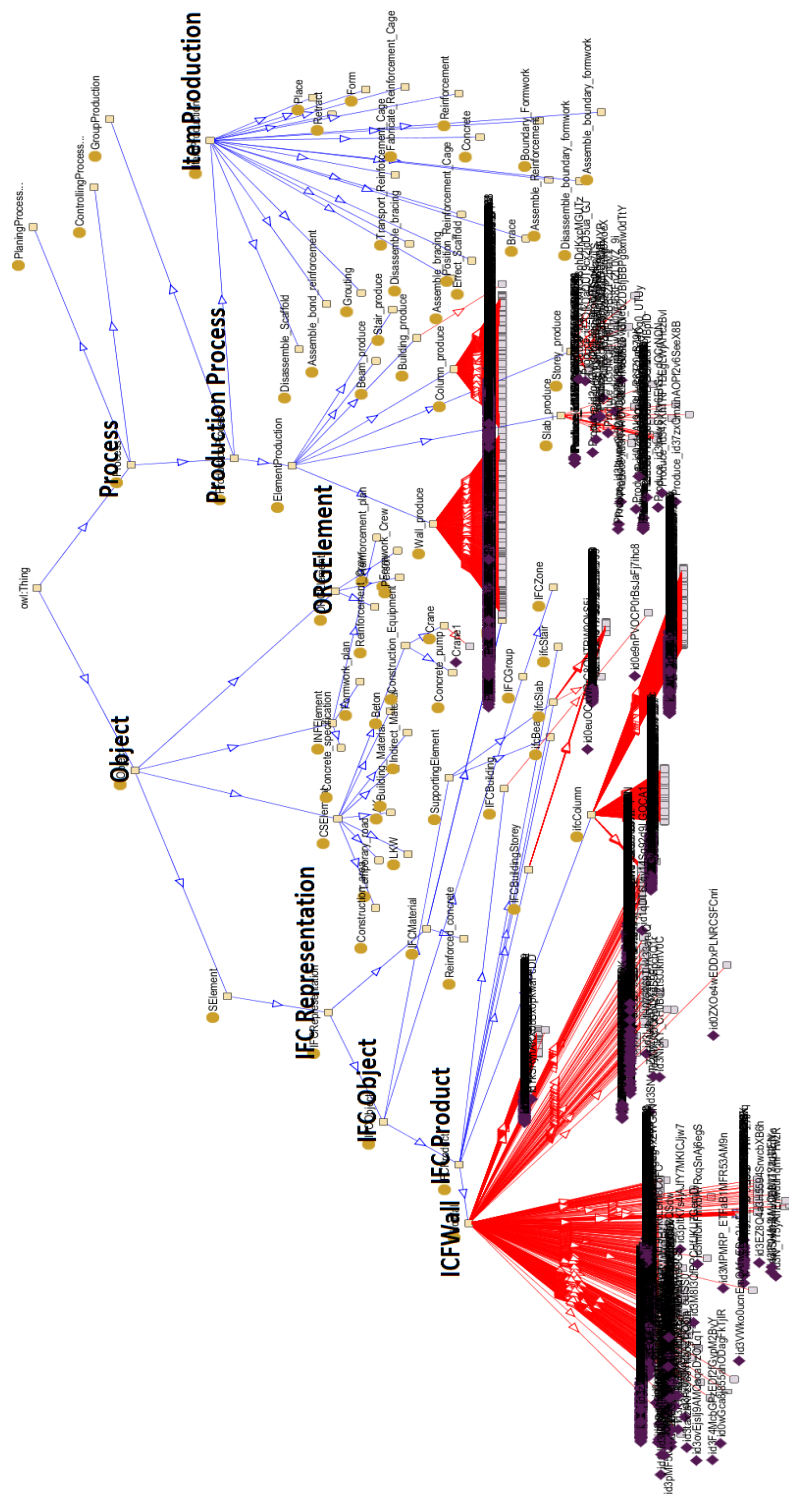


Figure 7-3: General object and process model with instances

In the next step the configuration of the general process model with the process patterns is carried out. At first a search for available pattern modules in the Process Pattern Ontology is performed and then selected patterns are used to configure specific construction processes that will be saved then in the Process Instance Ontology. The configuration process is performed for each storey sequentially. For the high-rise office building following process patterns were chosen from the Process Pattern Ontology:

- *Produce a column on-site;*
- *Produce a wall on-site;*
- *Produce a slab on-site.*

These process patterns were assigned to all elements of the building and for all patterns a crane was chosen as a preferred construction machine.

After the second step the complete hierarchy of processes is modelled, however not all sequential relations between groups of subprocesses (such as *hasNext* or *hasPrevious*) are fully represented. These missing relations can be set by the user manually (basic configuration) or chosen from the variants proposed by applying different configuration strategies in the next step.

The last step is the configuration of construction processes with the help of rules. In this step it is possible to apply different strategies in order to configure construction processes at each LoD. Two configuration strategies described in the Chapter 5 can be used for the current test case. The use of the Core and Rest configuration strategy (s. 5.3.2) can accelerate the construction process. In that case a special large formwork system for walls is used and therefore wall's elements from few different floors can be constructed in one procedure. This can reduce the construction time; however, the usage of the large formwork systems can increase costs of construction. For the model of the high-rise building both configuration strategies described in this work were tested.

A short comparison of the classic construction process, where each floors are constructed consequently and the construction after the *Core and Rest* configuration strategy (with parameter $P=2$) is shown in the Table 8. For the element construction the *Horizontal Matrix* method was chosen (elements are constructed from left to right according to their position).

It can be clearly seen that the second construction approach generates less workflow elements and groups, moreover the simulation shows that this approach can be faster in case a special large formwork system is used, however other parameters such as resource and material costs can be significantly higher in that case.

Table 8: Comparison of two configuration approaches for the case 1

| Use case 1 | | |
|---|---|---|
| Total number of elements before configuration | | 1135 |
| | Configuration 1 Floors: Construct consequently Elements: Horizontal Matrix | Configuration 2 Floors: Core-Rest strategy (P=2) Elements: Horizontal Matrix |
| Number of elements after configuration | 5672 | 5644 |
| Number of groups | 54 | 27 |

After the configuration phase a complete process workflow is saved in the Process Instance Ontology. This workflow is then exported in the BPMN format and can be used further for the simulation (Figure 6-1). In Figure 7-4 some fragments of the exported process workflow for the example of the high-rise office building are shown. On the left the fragment of the BPMN workflow for the storey level is shown. In the middle of the figure one of the processes of the storey level (*Produce elements of the Rest group* at the storey number five) is expanded and shows its different subprocesses representing the group level. The element level is presented at the right side of the figure, where one of the processes of the group level is expanded and shows the construction process of some particular column in that group. The presented results are obtained by application of the *Core and Rest* configuration strategy.

Having as input process workflow simulation software generates schedules and provides information about machines and material usage. Moreover, in the simulation process it is possible to consider different construction scenarios without changing the generated process workflow just by manipulating the number of different construction machines and construction teams. The detailed description of the simulation process as well as analysis of the simulation results lie out of the scope of this thesis and can be found in (Scherer & Schapke, 2014) (Ismail A., Benevolenskiy A., 2011).

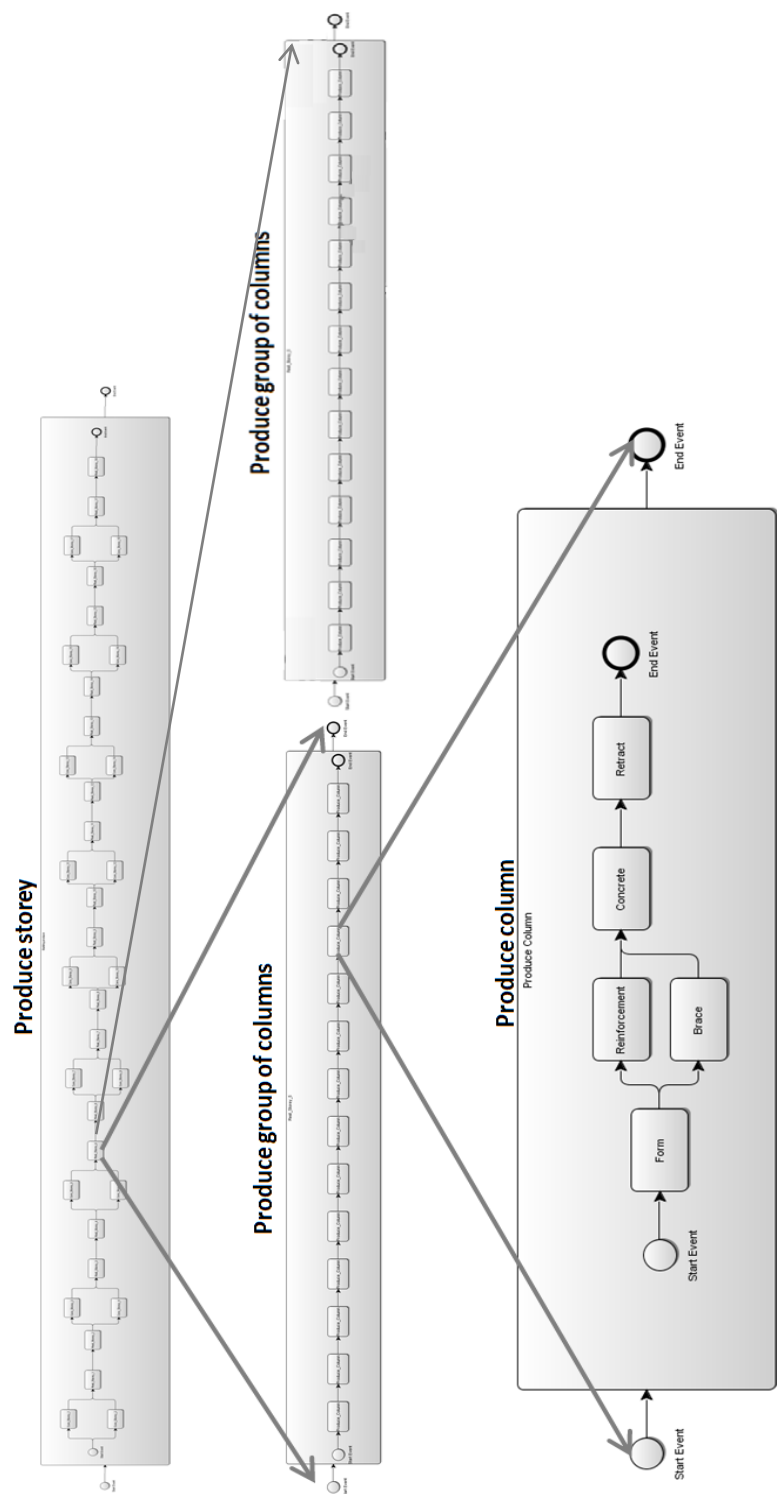


Figure 7-4: Fragments of the process workflow with different levels of details (LoD) (left to right: storey level, group level, element level)

7.2. Case 2

For the second case a model of a five-storey school building was chosen. In that case also only main structural elements such as walls, slabs and stairs were considered. In Figure 7-5 a 3D-visualization of the school building is shown.

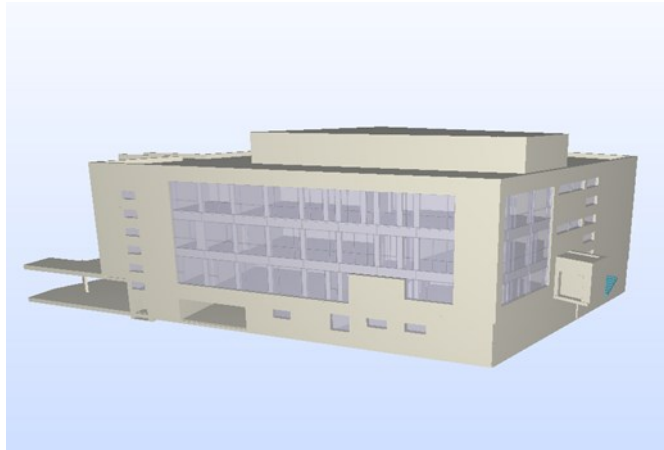


Figure 7-5: School building

The configuration process for this case is very similar to the one from the first example and consists of the same three steps shown in Figure 7-2. For the construction process of the school building for this example two different scenarios were considered. In the first case prefabricated construction elements were used and in the second case all elements were constructed on-site. Therefore some additional process patterns were used in that use case.

Following process patterns from the Process Pattern Ontology were chosen:

- *Produce a wall on-site;*
- *Produce a precast wall;*
- *Produce a slab on-site;*
- *Produce a precast slab;*
- *Produce a stair on-site;*
- *Produce a precast stair.*

In the configuration step, *Group and Sequencing* configuration strategy, partially described in chapter 5, was used. It allows generating groups of elements at different floors depending on the elements type and their position. After the grouping further configurations are performed within these groups.

A short comparison of two configuration approaches is shown in the Table 9. In the first approach process patterns for the precast construction were used. Process patterns for the on-site construction were used in the second configuration approach. The same configuration strategy was chosen for both approaches. As it can be seen the configuration approach with the process patterns for prefabricated elements generates fewer elements in the final process workflow, however that does not mean that this configuration approach is better or worse because the quantitative evaluation of the proposed workflows can be done only at the simulation phase, where information about resource usage, required time and estimated cost for each workflow task can be obtained and analyzed.

Table 9: Comparison of two configuration approaches for the case 2

| Use case 2 (Floor 0) | | |
|---|--|--|
| Total number of elements before configuration | | 185 |
| | Configuration 1 Precast construction Strategy: Group and sequencing | Configuration 2 On-site construction Strategy: Group and sequencing |
| Number of workflow elements after configuration | 1442 | 1954 |
| Number of groups | 5 | 5 |

Some parts of the exported fragments of the configured process workflow (Configuration 1) are shown in Figure 7-6. At first processes for construction of five floors are connected consequently. This workflow is shown on the left side of the figure. In the middle two process workflows are shown, which represent the grouping of all elements in some floor according to their type and the application of the configuration strategies. The process workflow for the element level is shown on the right side of the figure.

As in the first case, the exported in the BPMN XML process workflow can be visualized with any BMPN viewer (BPMN 2.0) or used as an input data for the simulation.

For all cases it is also important to notice, that the configuration in Process Configurator does not guarantee finding the best solution, but provides different variants of the workflows that can be simulated and analyzed later.

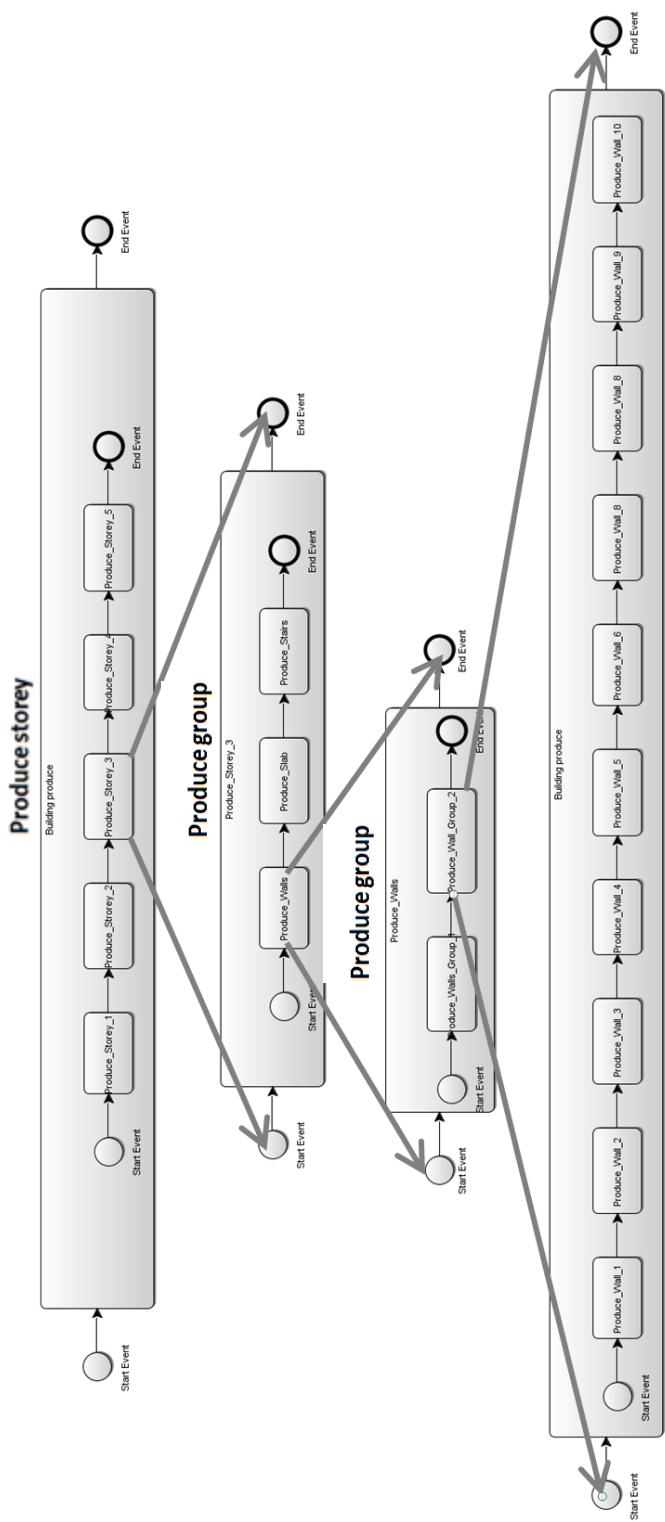


Figure 7-6: Fragments of the process workflows for the construction of the school building (left to right: storey level, group level, element level)

8. Conclusions and future research

This chapter concludes the thesis and offers an outlook on further research in the area of construction process modeling and configuration. Positive results as well as some limitations and "bottlenecks" of the current implementation are shown. In the last section a proposal for the new methodology of the general process configuration and modeling is introduced briefly.

8.1. Conclusions

The thesis introduced a new methodology for the modeling and configuration of construction processes. The novelty of this research work lays in the use of the ontology-based approach for the modeling of construction processes and its combination with the rule-based process configuration. The initial assumption was that the usage of generic process patterns allows simplifying and speeding up the development of the process overall schedules, which requires a lot of work and time. The goal of this work was to develop a formal high-level model for construction processes and process patterns and to provide a semi-automated mechanism for their configuration.

The proposed system consists of two main parts, which are developed independently of each other but work together closely. These parts are ontologies, which are used for the process modeling, and a rule-engine with sets of rules, which is used for the configuration.

The research efforts have been focused mainly on the formalization and modeling of construction processes. An important role in the considered methodology plays the ontological framework which includes the Process Pattern and Process Instance Ontologies. Process patterns stored in the Process Pattern Ontology are standardized construction processes that can be individually instantiated for many different construction processes. Process instances for specific construction projects are saved in the Process Instance Ontology. The instantiation process of this ontology is not trivial and consists of few steps, including the generation of the general models, their adaptation to the specific project conditions and the configuration with the application of various configuration strategies.

A prototype Java-based application called Process Configurator was implemented within this work. Currently, it provides the defined ontology specifications, an initial knowledge base of process patterns and a set of construction process rules for the subdomain "structural concrete works". Different configuration strategies were proposed and realized by means of hierarchical rule sets. The usage of such a rule-based approach

demonstrated highly flexible configurability of construction processes at every stage of its life-cycle.

Several use-cases were used to demonstrate the application of the approach proposed in this work. The obtained results were exported from the Process Instance Ontology and then transformed into a BPMN format in order to be easily presented to the end users in the form of BPMN process workflows or used further in the discrete-event simulation software. Differently configured process workflows for the same construction project resulted in diverse process schedules after the simulation. This supports the user in finding an optimal solution by applying various configuration strategies.

However, there are still a lot of topics to be investigated. Future development efforts are needed regarding the definition of new process patterns and the increase of the rule base. Currently, process patterns can be defined in the BPMN format by using some subsets of its elements, and then imported in the Process Pattern Ontology. The possibility to support the whole set of BPMN elements can make the integration with the BPMN format fully complete. Another objective is a further improvement in the integration of the rule-engine with the ontological knowledge base. As it was shown, homogeneous solutions are normally easier in the technical implementation, while hybrid solutions provide better reasoning support. The choice of the right approach is an important decision, which depends on the specific project goals and requirements, and therefore no general recommendation can be given for all projects. The limitation of the configuration approach can be the application of the various configuration strategies in different projects. Even though the configuration strategies are defined on a very general level, their application for specific and complex construction projects is not always trivial. Therefore, a possibility for the interactive modification of the strategy by users, depending on changing project conditions, can be a desirable further extension.

Finally, it is also necessary to note, that the developed methodology for the ontology-based process modeling and configuration is currently being further extended and developed in the research work of Mrs. Ksenia Roos, which should be published later as a separate doctoral thesis. The main focus of her research work lies on the description and comparison of the different configuration approaches and strategies and therefore a more detailed description of these issues is omitted in the current work.

8.2. Outlook

The approach proposed in the previous chapters shows how the ontology-based model can be used for the modeling and configuration of construction processes. The introduced methodology demonstrated its power and suitability for the area of construction

processes, where the structural concrete works are considered. However, there are still some important aspects that can be considered as potential research topics.

The proposed system is open to further expansions and creates a possibility of future research and development. A big research potential can provide a methodology for the general process configuration and modeling, which is introduced in the following section. This methodology is based on the concepts and methods already successfully implemented and described in the main part of this thesis, but it is modeled to be more general and so applicable for different domains.

A possible extension of the configuration approach proposed in the chapter 4 can be a process configuration method with preliminary costs and time estimations on different level of details. Both approaches are presented shortly in the following two sections.

An extension of the ontological framework with a possible integration of the ifcOWL ontology in it is considered in the last section

A methodology for the general process modelling and configuration

One of the main ideas of the proposed approach is to use a general and domain-independent structure for the process pattern. This general structure should consist of several main components such as: *Process*, *Resource*, *Description*, *Meta Data* and *Sub-processes* with their order. So the basic structure of the process pattern remains quite similar to the one, proposed in the chapter 4, however its subconcepts will not be specified explicitly. Therefore this general structure can be used to describe not only construction processes but also other type of production or information processes and can be quickly adopted for various projects and domains.

The patterns are formalized in the BPMN standard, which has an advantage that patterns can be easily visualized with the help of many BPMN tools and exchanged between different systems.

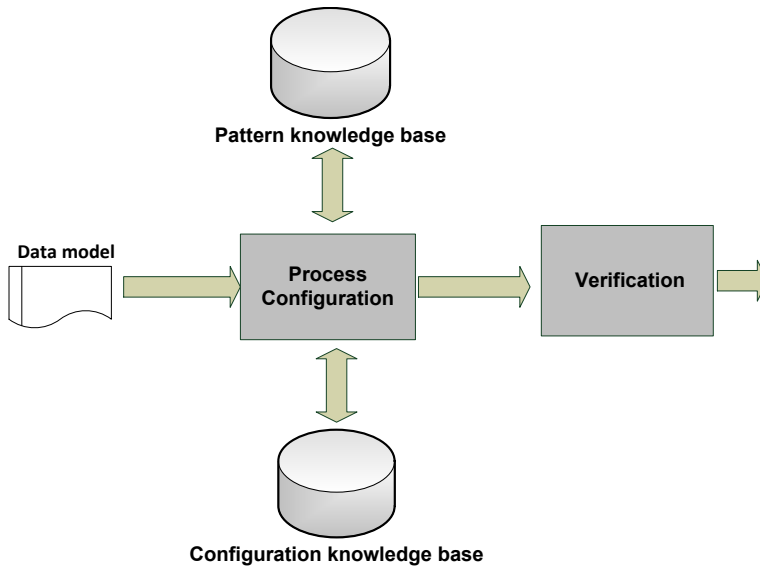


Figure 8-1: Main components of the system

An outline of the overall system is presented in Figure 8-1 and includes five main components that are interacting with each other:

- Data model represents the input data model for the system;
- Process configuration component is a core component of the system, which is used for the instantiation and configuration of the generalized process patterns;
- Configuration knowledge base is used to store configuration strategies and rules. The rules and strategies should operate only on the limited set of concepts described in the process patterns. This allows avoiding problems by application of the strategies in different projects, however, it limits their expressive power;
- Pattern knowledge base stores different process patterns and their application knowledge;
- Verification component is used for verification of the configured process modules with the help of the Petri nets. It can be used to identify and avoid such problems like deadlocks, infinite loops or some logical errors.

The introduced methodology can be considered as a possible future research topic, which can enhance and continue the research presented in this thesis.

Possible extension of the configuration approach

Most of the configuration strategies implemented in this work consider mainly technological and geometrical aspects of the configuration by using Building Information Model (BIM) as an input. However cost and time estimations are one of the most important

steps in construction project management and therefore they also should be taken into account in the future work. Furthermore, preliminary estimation of time and cost already at the early configuration steps can not only provide necessary information for planning of construction process, but also has a significant influence on the configuration process itself.

An interesting extension of the developed approach can be the integration of the initial estimating methods into the configuration process on different levels of detail, which allows getting a rough estimation of the required time and costs already in the early stages without performing a process simulation.

This approach allows coupling the estimation process with the configuration process. Moreover, the hierarchical structure of the process model provides not only possibilities for the top-down process configuration, but also a good basis for the calculation process. The idea to associate configuration and estimation processes comes from their nature. For both processes it is necessary to break down the initial process in a number of tasks, therefore the approach starts from the whole building and then details the processes until reaching the elements level.

Two main groups of parameters: cost related parameters and time related parameters are already defined in the structure of the process pattern presented in the chapter 4.

Two examples of the simple time and cost parameters are presented below:

- *Time parameter: Time_Value h/m³*
- *Cost parameter: Cost_Value Euro/m³*

However other appropriate basic values instead of m³ can be chosen. The first parameter allows estimating approximate amount of time in hours for producing one cubic meter of some construction element. The second parameter estimates approximate price in Euro for producing one cubic meter of the building element. For example having for a concrete wall a *Time_Value*=2 and a *Cost_Value*=135 it is easy to calculate that construction of a wall with 3 cubic meters volume will cost approximately 405 Euro and lasts at least 6 hours. Of course it is necessary to note that this estimation approach is very rough, because many important factors (such as used resources, complexity and location of the elements, the execution order of the processes and many others) are not considered and therefore this approach cannot be used for the precise project time and budget planning. However, this approach performed in parallel to the configuration allows obtaining not only different configuration variants, but also providing rough estimation of their approximate duration and costs values.

Extension of the ontological framework with ifcOWL

Process Pattern and Process Instance Ontologies developed in this work include different concepts and properties for describing construction processes, object, resources, relations between these elements and references to the elements from other domain models. For the current implementation, an Object model was developed, which taxonomy and concepts are partially presented in the chapter 4. Regarding the representation of this model in the ontological framework it is necessary to mention the ifcOWL, an ontology for building and construction sector based on the industry foundation classes proposed in (Beetz J. , 2009). Because of the fact that the initial development of the ontologies presented in this thesis began almost at the same time, when the first publication of the ifcOWL was available, it was not possible to consider the integration of that approach in the developed ontological framework at the beginning. However, now from the actual perspective we have a possibility to analyze this approach and discuss future possibilities of the integration of the ifcOWL in the current work.

Different transformations of the Express schemas into ontologies have been suggested in the last years. W3C Linked Building Data Community Group (LBDC-Group, 2014) and the BuildingSMART Linked Data Working Group (BuildingSMART, 2015) created in June 2014 bring together experts in BIM and the Web of Data technologies to define existing and future use cases and requirements for linked data based applications across the building life cycle. These groups maintain several resources, e.g. Linked Building Data Community Group²⁴ and Linked Data Working Group²⁵, that keep the documents related to the development of the ifcOWL ontology which can be published in-sync with IFC specification.

One of the approaches for transformation the IFC domain model into a Description Logic-based model was developed in the work of Beetz (Beetz J. , 2009). The transformation results are encoded in the Web Ontology Language OWL and can be populated with instances of existing building models. The resulting ifcOWL model covers the definition of the concepts from IFC specification as well as relations between them. For each language feature that is used in the modeling of IFC a respective transformation was proposed. In order to transform simple data types in EXPRESS special wrapper classes with a single owl:DatatypeProperty was used and therefore all attributes from IFC specification can be transformed consistently into owl:ObjectProperty.

As a possible extension of the current ontological framework presented in chapter 4, the substitution of the actual Object model with the subset of the ifcOWL ontology, describing the required for the current methodology set of concepts and relations can be con-

²⁴ <http://www.w3.org/community/lbd/ifcowl> Retrieved 2015-01-25

²⁵ <http://www.buildingsmart-tech.org/future/ifcowl> Retrieved 2015-01-25

sidered as a topic for further research. Even that the current implementation fulfills all necessary conditions and the transformation will require additional work in changing of the models and integration with rules, the new solution can provide a standardized and extendable Object model as well as improve and facilitate the future integration of the Process model with the BIM model.

9. Appendix A.

A.1. Process Pattern

Process pattern "Produce column with a crane" in RDF serialization

```

<owl:NamedIndividual rdf:about="http://www.mefisto-au.de/ontologies/ProcessOntology.owl# Column_produce1">
  <rdf:type rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl# Column_produce"/>
  <hasDescription rdf:datatype="&xsd:string">Produce column with crane</hasDescription>
  <hasContext rdf:datatype="&xsd:string">ifcColumn</hasContext>
  <hasContext>produce</hasContext>
  <hasContext>Column</hasContext>
  <hasSubTask rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Brace1"/>
  <hasSubTask rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Concrete1"/>
  <hasResource rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Crane1"/>
  <hasSubTask rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Form1"/>
  <hasSubTask rdf:resource="http://www.mefisto-
    bau.de/ontologies/ProcessOntology.owl#Reinforcement1"/>
  <hasSubTask rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Retract1"/>
  <hasObject rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#ifcColumn1"/>
</owl:NamedIndividual>

<!-- http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Form1 -->
<owl:NamedIndividual rdf:about="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Form1">
  <rdf:type rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Form"/>
  <hasNext rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Brace1"/>
  <hasNext rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Reinforcement1"/>
</owl:NamedIndividual>

<!-- http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Brace1 -->
<owl:NamedIndividual rdf:about="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Brace1">
  <rdf:type rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Brace"/>
  <hasNext rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Concrete1"/>
</owl:NamedIndividual>

<!-- http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Reinforcement1 -->

```

```

<owl:NamedIndividual rdf:about="http://www.mefisto-
    bau.de/ontologies/ProcessOntology.owl#Reinforcement1">
    <rdf:type rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Reinforcement"/>
    <hasNext rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Concrete1"/>
    <hasResource rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Crane1"/>
    <hasResource rdf:resource="http://www.mefisto-
        bau.de/ontologies/ProcessOntology.owl#Reinforcement1"/>
</owl:NamedIndividual>
<!-- http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Concrete1 -->
<owl:NamedIndividual rdf:about="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Concrete1">
    <rdf:type rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Concrete"/>
    <hasResource rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#AK1"/>
    <hasResource rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Beton1"/>
    <hasNext rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Retract1"/>
</owl:NamedIndividual>
<!-- http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Retract1 -->
<owl:NamedIndividual rdf:about="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Retract1">
    <rdf:type rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#Retract"/>
    <hasResource rdf:resource="http://www.mefisto-bau.de/ontologies/ProcessOntology.owl#LKW1"/>
</owl:NamedIndividual>

```

A.2. Rules overview

Table 10: Overview of rules

| Name | Level | Description | Implementation |
|--|----------------|--|--------------------------------|
| Build consequently | <i>Storey</i> | Construct storeys of the building consequently according to their elevation | <i>Roos/ Benevolenskiy</i> |
| Build core at first | <i>Storey</i> | Construct at first core elements of several storeys and then the rest elements (s. Core and Rest configuration strategy) | <i>Benevolenskiy</i> |
| Group core elements | <i>Group</i> | Group all core elements of the building in respective groups (s. Core and Rest configuration strategy) | <i>Benevolenskiy</i> |
| Group rest elements | <i>Group</i> | Group all rest elements of the building in respective groups (s. Core and Rest configuration strategy) | <i>Benevolenskiy</i> |
| Group by distance matrix | <i>Group</i> | Group elements according to the distance from each other (s. Group and sequencing configuration strategy) | <i>Benevolenskiy</i> |
| Construct walls/columns/group at first | <i>Group</i> | Construct elements of the selected group at first, other groups will be constructed afterwards (combinatory rule) | <i>Roos</i> |
| Build by Horizontal Matrix | <i>Element</i> | Build elements in the order according to their position in the storey (coordinate X) | <i>Roos/ Benevolenskiy</i> |

| | | | |
|--|----------------|--|--------------------------------|
| Build by Vertical Matrix | <i>Element</i> | Build elements in the order according to their position in the storey (coordinate Y) | <i>Roos/ Benevolenskiy</i> |
| Build by TSP algorithm | <i>Element</i> | Order the elements/processes in some group according to the Traveling Salesman Problem algorithm | <i>Roos</i> |
| Concrete/Task in parallel | <i>Item</i> | Execute concreting/other works in parallel, other tasks will be executed sequentially | <i>Roos</i> |
| Concrete/Task execute with parameter X | <i>Item</i> | Execute concreting/other task in one group of elements in parallel for the x number of elements | <i>Roos</i> |
| Build by priority | <i>Item</i> | Execute elements in one group according to the priority of each elements | <i>Roos/ Benevolenskiy</i> |

A.3. Rules implementation

In this section implementations of several rules developed in this work are presented. It is necessary to note here, that from the rules some external methods from the Process Configurator are called (for example *ProcessConfigurator.makeRest*).

Rule “Build consequently” in Drools syntax

```
global java.util.ArrayList all;

/*1*/rule "Storey_rule"
agenda-group "main"
no-loop true
//lock-on-active true

when
$so1: IFCElement($id1:ID, type=='ifcStorey', $e1:elevation)
$p1: Process(type=='Storey_produce', outputObjekt contains $id1)
$so2:IFCElement($id2:ID, type=='ifcStorey', elevation==($e1+1))
$p2:Process(type=='Storey_produce', outputObjekt contains $id2)

then
System.out.println("You can execute process "+$p2.ID+ " after the process " + $p1.ID+ " .");
ProcessConfigurator.dohasNext($p2.ID,$p1.ID);
$p1.next.add($p2);
update($p1);
end

/*2*/rule "output1"

salience 200
agenda-group "complementary"

when
$p1: Process(type=='Storey_produce')
not (exists ($p3:Process(type=='Storey_produce',next contains $p1)))

then
System.out.println("OK1" + $p1.ID);
all.add($p1.ID);
end

/*3*/rule "output2"
salience 100
agenda-group "complementary"
//no-loop true
//lock-on-active true

when
$p2: Process()
$p1:Process(next contains $p2)

then
if (all.contains($p1.ID)) {all.add($p2.ID);}
end
```


Rule “Group rest elements” in Drools syntax

```
package de.tudresden.bau.cib.OntologyMatcher;
import java.lang.Double;
global java.util.ArrayList all;

/*1*/rule "Find max y"
agenda-group "horizontal"
dialect "java"

when
$e1:IFCElement (y>=0)
not (exists ($e2:IFCElement (y>$e1.y)))

then
all.add($e1.ID);
retract($e1);
end

/*2*/rule "Find min x"
agenda-group "vertical"
dialect "java"

when
$e1:IFCElement (x>=0)
not (exists ($e2:IFCElement (x<$e1.x && x>0)))

then
all.add($e1.ID);
retract($e1);

end
```

Rule “Group core elements” in Drools syntax

```
package de.tudresden.bau.cib.OntologyMatcher;
import org.drools.planner.examples.tsp.ProcessConfigurator;

/*0*/ rule "Create Core-Groups and Core-Processes"
salience 500
//no-loop true
//lock-on-active true

when
$o2:IFCElement($id2:ID, type=='ifcWall')
$o1: IFCElement ($id1:ID, type=='ifcStorey', $e1:elevation, contains contains $o2)

then
ProcessConfigurator.makeKern($o1.ID,$o2.ID);

end
```

Rule “Group rest elements” in Drools syntax

```
package de.tudresden.bau.cib.OntologyMatcher;
import org.drools.planner.examples.tsp.ProcessConfigurator;

/*0*/ rule "Create Rest-Groups und Rest-Processes"
salience 500
//no-loop true
//lock-on-active true

when
$so2:IFCElement($id2:ID, type!='ifcWall')
$so1: IFCElement ($id1:ID, type=='ifcStorey', $e1:elevation, contains contains $so2)

then
ProcessConfigurator.makeRest($so1.ID,$so2.ID);

end
```

10. References

- Abanda, F., Tah, J., & Keivani, R. (2013). Trends in built environment semantic Web applications: Where are we today? *Expert Systems with Applications, Volume 40, Issue 14, 15 October 2013*, 5563–5577.
- Aguilar-Savén, R. (2004). Business process modelling: Review and framework. *International Journal of Production Economics, Volume 90, Issue 2, 28 July 2004, Pages 129–149*.
- Ambler, S. W. (2010). *The Object Primer: Agile Model-Driven Development with UML 2.0*. Cambridge University Press (Virtual Publishing).
- Archibald, R., & Villoria, R. (1967). *Network-Based Management Systems (PERT/CPM)*. John Wiley & Son.
- Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. (2003). *Description Logic Handbook: Theory, Implementations, and Applications*. Cambridge: Cambridge Univ. Press.
- Bali, M., Taylor, J., & Larbi, S. (2009). *Drools JBoss Rules 5.0 developer's guide*. Birmingham, UK: Packt Pub.
- Banks, J. (2009). *Discrete-Event-Simulation*. Prentice Hall; 5 edition.
- Beetz, J. (2009). *Facilitating distributed collaboration in the AEC/FM sector using Semantic Web Technologies* (Phd Thesis Ausg.). Eindhoven, The Netherlands: TU Eindhoven, Design Systems Group, Prof. B. de Vries.
- Beetz, J., van Leeuwen, J. P., & de Vries, B. (2009). IfcOWL: A case of transforming EXPRESS schemas into ontologies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*(23, Special Issue 01), S. 89-101.
- Beißert U., König M., Bargstädt H –J. (2007). Constraint-Based Simulation of Outfitting Processes in Building Engineering. *Proc. 24th International Conference Managing IT in Construction CIB W078, Maribor, Slovenia*.
- Benevolenskiy A., Katranuschkov P. & Scherer R. (2011). Ontology-based configuration of construction processes using process patterns. *Proceedings of the 2011 EG-IC Workshop, Enschede, The Netherlands*.
- Benevolenskiy, A., Roos, K., & Scherer, R. J. (2012). Using Rules for the Configuration of Construction Processes. *Proceedings of the ICCCBE 2012, Moscow, Russia, June 2012*.
- Benevolenskiy, A., Roos, K., Katranuschkov, P., & Scherer, R. (2012). Construction processes configuration using process patterns. *Advanced Engineering Informatics, Volume 26, Issue 4, 727–736*.
- Benevolenskiy, A., Roos, K., Scherer, R., & Katranuschkov, P. (2014). Ontologiebasiertes Framework für Referenzprozesse und Prozesskonfiguration. In R. Scherer, & S.-E. Schapke (Hrsg.), *Informationssysteme im Bauwesen 1* (S. 273-287). Springer.

- Bergman, M. (2009). *The Open World Assumption: Elephant in the Room*. Retrieved 12 1, 2014, from AI3::Adaptive Information: <http://www.mkbergman.com/852/the-open-world-assumption-elephant-in-the-room/>
- Berkhahn, V., Klinger, A., Rüppel, U., Meißner, U. F., Greb, S., & Wagenknecht, A. (2005). Processes Modelling in Civil Engineering based on Hierarchical Petri Nets. *CIB-W78, Dresden 2005*.
- Boehm, B. (1988). A Spiral Model of Software Development and Enhancement. *IEEE Computer* 21(5).
- Booch, G., Rumbaugh, J., & Jacobson, I. (1998). *Unified Modeling Language User Guide*. Addison Wesley.
- BuildingSMART. (2015). *BuildingSMART Linked Data Working Group*. Von <http://www.buildingsmart-tech.org/future/ifcowl> abgerufen
- Carnaghan, C. (2006). Business process modeling approaches in the context of process level audit risk assessment: An analysis and comparison. *International Journal of Accounting Information Systems, Volume 7, Issue 2, June 2006, Pages 170–204*.
- Castano, S., De Antonellis, V., Fugini, M. G., & Pernici, B. (1998). Conceptual schema analysis: techniques and applications. *ACM Trans. Database Syst.* 23, 3 (September 1998), 286-333.
- Daconta, M., Obrst, L., & Kevin, S. (2003). *The Semantic Web*. Indianapolis, Ind: Wiley.
- Davenport, T. (1993). *Process Innovation: Reengineering work through information technology*. Harvard Business School Press.
- Davis, R. (2001). *Business process modelling with ARIS: A practical guide*. London: Springer.
- Defense Dept., Defense Acquisition University. (2001). *Systems Engineering Fundamentals*. Defense Acquisition University Press.
- Dolenc, M., Katranuschkov, P., Gehre, A., Kurowski, K., & Turk, Z. (2007). The IntelliGrid Platform for Virtual Organisations Interoperability. *ITcon Vol. 12, pp. 459-477*.
- eeEmbedded. (2015). *Collaborative Holistic Design Laboratory and Methodology for Energy-Efficient Embedded Buildings*. Von <http://141.30.165.10/> abgerufen
- El Kharbili, M., & Pulvermüller, E. (2009). A Semantic Framework for Compliance Management in Business Process Management. In *proceeding of: Business Process, Services Computing and Intelligent Service; Management, Leipzig, Germany*.
- El-Diraby, T., Lima, C., & B., F. (2005). Domain Taxonomy for Construction Concepts: Toward a Formal Ontology for Construction Knowledge. *Journal Comput. Civ. Eng.*, 19(4), 394–406.
- Enge, F. (2010). *Muster in Prozessen der Bauablaufplanung* (Phd Thesis, TU Berlin, Institut für Bauingenieurwesen, Prof W.Huhnt Ausg.). Shaker Verlag.

- Enge, F., & Huhnt, W. (2008). Optimal Component Types for the Design of Construction Processes. *12th International Conference on Computing in Civil and Building Engineering & 2008 International Conference on Information Technology in Construction*, 13.2008, No. S1.
- Eriksson, H.-E., & Penker, M. (1998). *Business Modelling with UML: Business Patterns at*. New York, NY, USA: Work John Wiley & Sons, Inc.
- Faschingbauer, G. (2011). *Simulationsbasierte Systemidentifikation im Rahmen der baubegleitenden geotechnischen Überwachung*. Dresden: Phd Thesis, Berichte des Insituts für Bauinformatik, Heft 8, Prof. R. Scherer.
- Fettke, P., & Loos, P. (2003). Classification of reference models: a methodology and its application. *Information Systems and e-Business Management; Volume 1, Issue 1*, pp 35-53.
- Fettke, P., & Loos, P. (2005). Der Beitrag der Referenzmodellierung zum Business Engineering,. *HMD* 241, 42. Jahrgang.
- Filkenstein, A., Kramer, J., & Goedicke, M. (1990). ViewPoint Oriented Software Development. *"Le Génie Logiciel et ses Applications"*, (S. 337-351). Toulouse.
- Fischer M., Aalami F. (1995). Scheduling with Computer-Interpretable Construction Method Models. *CIFE Working Paper, Center for Integrated Facility Engineering, Stanford, CA*.
- Forgy, C. (1982). Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence*, S. 17-37.
- Fowler M. (2004). *UML Distilled: A Brief Guide To The Standard Object Modelling Language*, Longman. Boston, Massachusetts: Addison-Wesley, 3rd edition.
- Franz, V. (2010). Unikatprozesse und ASIM-Aktivitäten. *Bericht von der Arbeitsgruppe „Unikatprozesse“*. In: *Modellierung von Prozessen zur Fertigung von Unikaten*. Weimar, Germany, 5-16.
- Gailly, F., & Poels, G. (2007). Ontology-driven Business Modelling: Improving the Conceptual Representation of the REA Ontology. *Working Papers of Faculty of Economics and Business Administration, Ghent University, Belgium*.
- Gali, A., Chen, C., Claypool, K., & Uceda-Sosa, R. (2005). From ontology to relational databases. *Conceptual Modeling for Advanced Application Domains, LNCS, vol 3289*, pp 278–289.
- Gasevic, D., & Devedzic, V. (2007). Interoperable Petri net models via ontology. *International Journal of Web Engineering and Technology; Volume 3 , Issue 4*,pp. 374-396.
- Genesereth , M., R., F., R., B., T., G., P., H., R., L., et al. (1992). *Knowledge Interchange Format Version 3.0 Reference Manual*. Stanford: Stanford University.
- Ghidini, C., Rospocher, M., & Serafini, L. (2008). A formalisation of BPMN in description logics. *Technical Report TR 2008-06-004, FBK-irst*.

- Gomez-Pérez, A., Fernandez-Lopez, M., & Corcho, O. (2004). *Ontological Engineering*. Springer.
- Halpin, D. W., & Riggs, L. S. (1992). *Planning and Analysis of Construction Operations*. John Wiley and Sons Ltd.
- Henderson-Sellers, B., & Edwards, J. (1990). The Object-oriented Systems Life-Cycle,. *ACM 33, 9 (September 1990), 142-159*.
- Huhnt W. (2005). Generating Sequences of Construction Tasks. *Proceedings of 22nd of W78 Conference on Information Technology in Construction, p. 17-22, Dresden, Germany, 2005*.
- Huhnt, W., & Enge, F. (2006). Can algorithms support the specification of construction schedules? *ITcon Vol. 11, Special Issue Process Modelling, Process Management and Collaboration*, S. 547-564.
- Huhnt, W., Richter, S., Wallner, S., Habashi, T., & Krämer, T. (2010). Data management for animation of construction processes. *Journal of Advanced Engineering Informatics, 24(4)*, 404–416.
- Huhnt, W., Richter, S., Wallner, S., Habashi, T., & T., K. (2010). Data management for animation of construction processes. *Advanced Engineering Informatics, Volume 24, Issue 4, November 2010, Pages 404–416*.
- Humphrey, W. (1989). *Managing the Software Process*. Addison-Wesley.
- ISES. (2014). *Intelligent Services For Energy-Efficient Design and Life Cycle Simulation*. Retrieved 10 1, 2014, from <http://ises.eu-project.info>
- Ismail A., Benevolenskiy A. (2011). Simulation of construction variants of a high-rise building. (In German). *Scherer R.J, Tauscher H. & Schapke S.-E. (Eds.) Mefisto: Management-Führung-Information-Simulation im Bauwesen. Tagungsband 2. Kongress, Dresden*.
- Jernigan, F. (2007). *BIG BIM little bim*. Salisbury, USA: 4Site Press.
- Jongeling, R. (2006). *A process model for work-flow management in construction*. Lulea: Phd Thesis, Lulea University of Technology, Division of Structural Engineering, Prof. T.Olofsson.
- Kadolsky, M., Baumgärtel, K., & Scherer, R. (2014). An Ontology Framework for Rule-Based Inspection of eeBIM-Systems. Prague, Czech Republic: In: Proc. Creative Construction Conference 2014.
- Karhu, V. (2000). Proposed new method for construction process modelling. *International Journal of Computer Integrated Design and Construction, Vol. 2 No 3, pp. 166-182*.
- Karhu, V. (2001). A view-based approach for construction process modelling. *Journal of Computer-Aided Civil and Infrastructure Engineering 18, pp. 275-285*.
- Katranuschkov, P., Gehre, A., Keller, M., Schapke, S.-E., & Scherer, R. J. (2006). Ontology-Based Reusable Process Patterns for Collaborative Work

- Environments in the Construction Industry. P. Cunningham & M. Cunningham (eds.): *Exploiting the Knowledge Economy*, IOS Press, pp. 1055-1063.
- Keller, G., Nüttgens, N., & Scheer, A. (1992). Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK),. *Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89 (in German)*, University of Saarland, Saarbrücken, 1992.
- Kifer, M., & Lausen, G. (1989). F-logic: a higher-order language for reasoning about objects, inheritance, and scheme. New York, NY, USA: Proceedings of the 1989 ACM SIGMOD international conference on Management of data.
- Kindler, E. (2006). On the semantics of EPCs: Resolving the vicious circle. *Journal of Data & Knowledge Engineering*, vol. 56, no. 1, pp. 23–40.
- Klinger, A., Berkhahn, V., & König, M. (2006). Formal treatment of additions in planning processes. *Proceedings of XIth International Conference on Computing in Civil and Building Engineering*, (S. 2883-2891). Montreal, Vanada.
- Koschmider, A., & Ried, D. (2005). Semantic annotation of Petri nets. *Proc. of AWPn*, S. 66-71. Schmidt, K.; Stahl, Chr.(Eds), *Informatik-Berichte, Humboldt-Universität Berlin*.
- LBDC-Group. (2014). *ifcOWL ontology*. Von <http://www.w3.org/community/lbd/ifcowl/> abgerufen
- McGuinness, D., & van Harmelen, F. (2004). *OWL Web Ontology Language*. Abgerufen am 15. 2 2012 von <https://www.w3.org/TR/owl-features/>
- Mikulakova, E., König, M., Tauscher, E., & Beucke, K. (2010). Knowledge-based schedule generation and evaluation. *Advanced Engineering Informatics*(24), S. 389-403.
- Mittal, S., & Frayman, F. (1989). Towards a Generic Model of Configuration Tasks. *Proc. 11th Int'l Joint Conference Artificial Intelligence*, Morgan Kaufmann, San Francisco, 1989, pp. 1395-1401.
- Moonseo, P., Kyung-won, L., Hyun-soo, L., Pan, J., & Jungho, Y. (2013). Ontology-based construction knowledge retrieval system. *KSCE Journal of Civil Engineering*, November 2013, Volume 17, Issue 7, pp 1654-1663.
- Noguera, M., Chung, L., Garrido, J. L., Hurtado, M. V., & Rodriguez, M. L. (2010). Ontology-driven Analysis of UML-Based Collaborative Processes using OWL-DL and CPN. *Journal of Science of Computer Programming*, vol 75, no. 8 , pp. 726-760.
- Osterwalder, A. (2004). *The Business Model Ontology – A Proposition in a Design Science Approach*. Ph.D. thesis, University of Lausanne, Ecole des Hautes Etudes Commerciales HEC: 173, Prof. Y. Pigneur.
- Ould M. A. (1995). *Business processes* (Repr Ausg.). Chichester: Wiley.
- Pahl, P., & Damrath, R. (2001). *Mathematical foundations of computational engineering: a handbook*. Berlin: Springer.

- Pan, J. (2006). *Construction project information management in a semantic web environment*. PhD thesis, Loughborough University, Loughborough, Prof. C. Anumba.
- Petri, C. A. (1962). Fundamentals of a Theory of Asynchronous Information Flow. *IFIP Congress 1962*: 386-390.
- Phalp, K. (1998). CAP framework for business process modeling. *Information and Software Technology*, 40 (13) , S. 731-744.
- Potts, C. (1989). A Generic Model for Representing Design Methods. *Proceedings of the 11th International Conference on Software Engineering*.
- Recker, J., Rosemann, C., Indulska, M., & Peter, M. (2009). Business process modeling : a comparative analysis. *Journal of the Association for Information Systems*, 10(4), pp. 333-363.1.
- REWERSE. (2008). *Reasoning on the Web with Rules and Semantics*. Von <http://rewerse.net> abgerufen
- Rolland, C. (1998). A Comprehensive View of Process Engineering. *Proceedings of the 10th International Conference CAiSE'98. B. Lecture Notes in Computer Science 1413, Pernici, C.Thanos (Eds), Springer. Pisa, Italy*.
- Rolland, C., Nurcana, S., & Grosza, G. (1999). Enterprise knowledge development: the process view. *Journal of Information & Management Volume 36, Issue 3*, 165-184.
- Roos, K., Benevolenskiy, A., & Scherer, R. (2014). Towards Knowledge-based Process Configuration in Construction. Cardiff: EG-ICE Workshop 2014.
- Roos, K., Benevolenskiy, A., Katranuschkov, P., & Scherer, R. (2014). Wissensbasierte Prozesskonfiguration von Bauprozessen. In R. Scherer, & S.-E. Schapke, *Informationssysteme im Bauwesen 1* (S. 289-309). Springer.
- Rosemann, M. (2003). *Application reference models and building blocks for management and control*. Handbook on enterprise architecture : with 23 tables.
- Rosenkranz, F. (2005). *Geschäftsprozesse: Modell- und computergestützte Planung*, 2. Auflage. Springer.
- Royce , W. (1970). Managing the Development of Large Software Systems. *Technical Papers of Western Electronic Show and Convention WesCon*.
- RuleML. (2002). *Rule Markup Language*. Von www.ruleml.org abgerufen
- Sabin, D., & Weigel, R. (1998). Product configuration frameworks – A survey. *IEEE Intelligent Systems (1998)*, pp. 42–49.
- Salimifard, K., & Wright, M. (2001). Petri net-based modelling of workflow systems: An overview . *European Journal of Operational Research, Volume 134, Issue 3, 1 November 2001, Pages 664–676*.
- Scheer, A.-W. (2000). *ARIS - Business Process Modeling*. Berlin, Germany: Springer, 3rd edition.

- Scherer R.J., S.-E. Schapke. (2011). A distributed multi-model-based management information system for simulation and decision-making on construction projects. *Journal of Advanced Engineering Informatics*, vol. 25, pp. 582-599.
- Scherer, R. J., Katranuschkov, P., & Rybenko, K. (2008). Description Logic Based Collaborative Process Management. In Y. Rafiq, P. de Wilde, & M. Borthwick (Hrsg.), *ICE08 – Proc. 15th workshop of the European Group for Intelligent Computing in Engineering (EG-ICE)*, (S. 291-302). Plymouth, UK.
- Scherer, R. J., Schapke, S.-E., & Katranuschkov, P. (2010). Concept of an information framework for management, simulation and decision making in construction projects. In K. Menzel, & R. J. Scherer (Hrsg.), *eWork and eBusiness in Architecture, Engineering and Construction - Proceedings of the* (S. 97-104). Cork, Ireland: Taylor & Francis Group.
- Scherer, R., & Schapke, S. (2014). *Informationssysteme im Bauwesen 1* (VDI-Buch Ausg.). Springer Berlin Heidelberg: Springer.
- Scherer, R., & Sharmak, W. (2008). Generic Process Template Description for the Effect of Risks on Project Schedule. ISBN 978-956-319-361-9. Santiago de Chile: CIB-W78, 25th International Conference on IT. ISBN 978-956-319-361-9, pp. 134-142.
- Scherer, R., Katranuschkov, P., & Rybenko, K. (2008). Description Logic Based Collaborative Process Management. Rafiq Y., de Wilde P. & Borthwick M. (eds.) *“ICE08 – Proceedings of the 15th Workshop of the European Group for Intelligent Computing in Engineering (EG-ICE)”*, Plymouth, UK, pp. 291-302.
- Sharmak, W. (2011). *Dynamic network planning in construction projects using configurable reference process models*. Dresden: Phd Thesis, Berichte des Instituts für Bauinformatik, Heft 9, Prof R. Scherer.
- SUPER Project. (2010). *Semantics Utilised for Process Management within and between Enterprises*. Retrieved 2 12, 2011, from <http://www.ip-super.org/>
- Tanyer, A., & Aouad, G. (2005). Moving beyond the forth dimension with an IFC-based single project database. *Automation in Construction*, Volume 14, Issue 1, January 2005, Pages 15–32.
- Tauscher, E. (2011). *Vom Bauwerksinformationsmodell zur Terminplanung - Ein Modell zur Generierung von Bauablaufplänen*. Weimar: Dissertation, Prof. Dr.-Ing. Beucke.
- Tauscher, E., Mikulakova, E., König, M., & Beucke, K. (2007). Generating Construction Schedules with Case-Based Reasoning Support . (S. 119-126). Pittsburgh, Pennsylvania, United States: Computing in Civil Engineering.
- Thomas, O., & Fellmann, M. (2009). *Semantic Process Modeling – Design and Implementation of an Ontology-based Representation of Business Processes*. (1. Aufl Ausg., Bd. v.No. 21). s.l: Springer-Verlag.
- Turk Z. (2006). Construction informatics: Definition and ontology. *Journal of Advanced Engineering Informatics*, vol. 20, no. 2, pp. 187-199.

- van Rees, R. (2006). *New instruments for dynamic building-construction*. PhD Thesis, TU Delft, Faculty of Civil Engineering and Geosciences.
- Vom Brocke, J. (2003). *Referenzmodellierung: Gestaltung und Verteilung von Konstruktionsprozessen* (Advances in Information Systems and Management Science Ausg.). Logos Berlin; Auflage: 1.
- Vysniauskas, E., & Nemuraite, L. (2006). Transforming ontology representation from owl to relational database. *Inf Technol Control* 35(3A):pp.333–343.
- W3C. (2004, February 10). *OWL Web Ontology Language Reference, W3C Recommendation*. Retrieved 11 14, 2012, from <http://www.w3.org/TR/owl-ref/>
- W3C Working Group. (2004, 05 21). *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. Retrieved 08 18, 2012, from <https://www.w3.org/Submission/SWRL/>
- Wang, H.-H., Boukamp, F., & T., E. (2010). An ontology-based approach to context representation and reasoning for managing context-sensitive construction information. *Journal of Computing in Civil Engineering*, vol. 25, no 5.
- WfMC. (1996). *The Workflow Management Coalition Specification*. Retrieved 3 14, 2013, from www.wfmc.org
- White, S. (2004). *Business Process Modeling Notation, specification of BPMN v1.0*.
- Wilcox, P., & Gurau, C. (2003). Business modelling with UML: the implementation of CRM systems for online retailing. *Journal of Retailing and Consumer Services*, Volume 10, Issue 3, May 2003, Pages 181–191.
- Wilson, R. (1979). *Introduction to graph theory*. Essex, England: Longman.
- Wu I.-C., Borrmann A., Beißert U., König M. & Rank E. (2010). Bridge construction schedule generation with pattern-based construction methods and constraint-based simulation. *Journal Advanced Engineering Informatics*, vol. 24, no. 4, pp. 379-388.
- Yabuki, N., & Shitani, T. (2005). A Management System for Cut and Fill Earthworks Based on 4D CAD and EVMS. *Computing in Civil Engineering (2005)*: pp. 1-8.
- Zalta, E. (2013). *The Stanford Encyclopedia of Philosophy*. Stanford: Stanford University.
- Zhiliang, M., Zhenhua, W. W., & L., Z. (2011). Application and extension of the IFC standard in construction cost estimating for tendering in China. *Automation in Construction*, Volume 20, Issue 2, March 2011, Pages 196-204.

Bisher erschienene Dissertationen, Habilitationen und Hefte des Instituts für Bauinformatik²⁶

| | | |
|---------------------|--|---------------------------------|
| Christoph Meinecke | Ein objektorientiertes Konstruktions-expertensystem mit Regelmethoden | Selbstverlag Karlsruhe, 1995 |
| Markus Hauser | Eine kognitive Architektur für die wissensbasierte Unterstützung der frühen Phasen des Entwurfs von Tragwerken | logos Verlag Berlin, 1998 |
| Martin Zsohar | Stochastische Größen der Resonanzfrequenzen und der Verstärkung seismischer Wellen im horizontal geschichteten zufälligen Medium | Shaker Verlag Aachen, 1998 |
| Christian Steurer | Modellierung und Berechnung des Ermüdungsrißfortschritts mit stochastischen Differential-gleichungen | Selbstverlag, 1999 |
| Peter Katranuschkov | A Mapping Language for Concurrent Engineering Processes | Heft 1, 2001 |
| Karsten Menzel | Methodik zur nachhaltigen, rechnergestützten Ressourcenverwaltung im Bauwesen | Heft 2, 2003 |
| Michael Eisfeld | Assistance in Conceptual Design of Concrete Structures by a Description Logic Planner | Heft 3, 2004 |
| Matthias Weise | Ein Ansatz zur Abbildung von Änderungen in der modellbasierten Objektplanung | Heft 4, 2006 |
| Jörg Bretschneider | Ein wellenbasiertes stochastisches Modell zur Vorhersage der Erdbebenlast | Heft 5, 2006 |
| Martin Keller | Informationstechnisch unterstützte Kooperation bei Bauprojekten | Heft 6, 2007 |
| Kamil Umut Gökce | IT Supported Construction Management | Heft 7, 2008 |

²⁶ Bis September 2003 Lehrstuhl für Computeranwendung im Bauwesen

| | | |
|----------------------|---|---------------|
| Gerald Faschingbauer | Simulationsbasierte Systemidentifikation im Rahmen der baubegleitenden geotechnischen Überwachung | Heft 8, 2011 |
| Wael Sharmak | Dynamic Network Planning in Construction Projects using Configurable Reference Process Models | Heft 9, 2011 |
| Amin Zahedi Khameneh | Wave-type based Real-Time Prediction of Strong Ground Motion | Heft 10, 2012 |
| Sebastian Fuchs | Erschließung domänenübergreifender Informationsräume mit Multimodellen | Heft 11, 2015 |