# Ontology-Based Configuration of Construction Processes Using Process Patterns

A. Benevolenskiy, P. Katranuschkov & R.J. Scherer
Dresden University of Technology, Germany
Alexander.Benevolenskiy@tu-dresden.de

**Abstract.** Process modelling is used in construction 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 needed each time to manually develop a project's overall process schedule. However, the total individual process is typically structured in multiple stages containing a number of recurring similar sub-processes. Such sub-processes can be represented and stored as generic reusable process patterns that can be standardized and instantiated for many different projects and processes. In this paper we describe the development of a formal high-level model for construction processes and a methodology for using process patterns in the configuration and execution of complex construction tasks. The paper discusses the ontology-based model and the use of process patterns and rules in the configuration of construction processes. The reported research is largely done in the frames of the German project Mefisto (2009-2012).

## 1 Introduction

Modelling plays a significant role in representing and describing complex construction processes on more abstract level. In the construction industry, process modelling is used to support simulation and estimate and plan required resources and costs. In the last years significant work has been done on the adaptation of Description Logic (DL) knowledge bases for process modelling. DL provides a logical formalism for the development of an ontology, providing a formal representation of knowledge as a set of concepts within a domain. The main advantage of the ontology use for the process formalisation is the possibility to analyse the obtained construction process with the help of different reasoning and consistency checking mechanisms. Early results achieved in the EU project InteliGrid (Katranuschkov at al. 2007) and the German BauVOGrid project (Scherer et al. 2008) have provided promising perspectives for further development. The proposed executable "*Business Process Objects*", representing reusable process patterns, could be used for the process instantiation and configuration.

The approach to generate construction schedules using reusable process patterns is not in principle new. Significant work has been already done in this area. An approach that considers generating construction schedules with case-based reasoning support is presented in (Tauscher et al. 2007). The methodology which assists in automating the generation of time schedules with pattern-based construction methods is considered within the ForBAU project (Wu et al. 2010). However, there are still little known efforts in the interactive ontology-based configuration of construction process using process patterns. The process patterns considered in this research, in contrast to the most time-scheduling approaches, encompass not only processes, but also resources and actors that play an important role in process planning and execution. The proposed methodology for the ontology-based process modelling is partially built upon the concepts suggested in (Katranuschkov et al. 2007).

In this paper we present the main concepts of an approach that allows interactive configuring of the construction process, depending on the actual resource availability, construction technique or some specific user requirements. Moreover, the significant influence on the

configuration process has a strategic knowledge, which refines processes in accordance to an overall plan. The designed system should support semi-automatic process generation and configuration. The presented research is part of the ongoing German research project Mefisto (2009-2012), which aims to develop a Management Information System to support decision making on different management levels in construction projects (Scherer et al. 2010).

## 2 Background

Our work is based on ontologies formalized in OWL (W3C 2004), the Web Ontology Language based on Description Logics (DLs). DLs are a family of knowledge representation formalisms that can be used to represent and reason about classes of individuals and relations between such classes in a formally understood way (Baader et al. 2003).

Nowadays ontologies are widely used. Originally the term "ontology" comes from philosophy, where it aims to represent the world using its objects and connections between them. 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. 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.

An ontology typically describes concepts (classes), instances (individuals) and relations between them. Of course the concrete realization depends on the language in which ontology is expressed, but these elements are common for all realizations. Usually relations in an ontology are represented through the hierarchy of classes, but they are not restricted just by them. Ontology is encoded using ontology description languages. Currently there are three main languages that can be used for that purpose: XML, RDF and OWL. All of them have different expressive power but have well-defined syntax, which makes them processable by computers. A good introduction to Semantic Web concepts and an overview of the ontology languages can be found in (Antoniou & Van Harmelen. 2004). A short comparison of these languages is presented in Table 1 below.

Table 1. Comparison of the expressive power of the different description languages

| | XML | XML Schema | RDF(S) | OWL |
|---|---|---|---|---|
| Class expressions | | | | + |
| Cardinality constraints | + | + | | + |
| Data types | | + | + | + |
| Enumerations | + | + | | + |
| Equivalence | | | | + |
| Extensibility | | | + | + |
| Inheritance | | | + | + |
| Formal semantics | | | + | + |

## 3 System Architecture

The main idea of the proposed approach is in the development and use of two separate but inter-related ontologies, i.e. (1) a *Process Pattern Ontology* and (2) a *Process Instance Ontology*, so that the process patterns from the first ontology can be used in many different construction processes represented by specific instantiations of the second ontology. 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 and includes a rule-based inference engine. In our work we consider additionally the possibility to use external Java-based rule engines, such as Drools (Bali 2009) or Jess (Friedman-Hill 2003), however this requires a transformation of the ontological description of the process into the appropriate rule-engine format. The ontological models are queried through SPARQL, an RDF query language. Pattern retrieval is also realized with the help of SPARQL, enabling the dynamic binding of search attributes/concepts.

The suggested overall system architecture is shown on Figure 1 below.

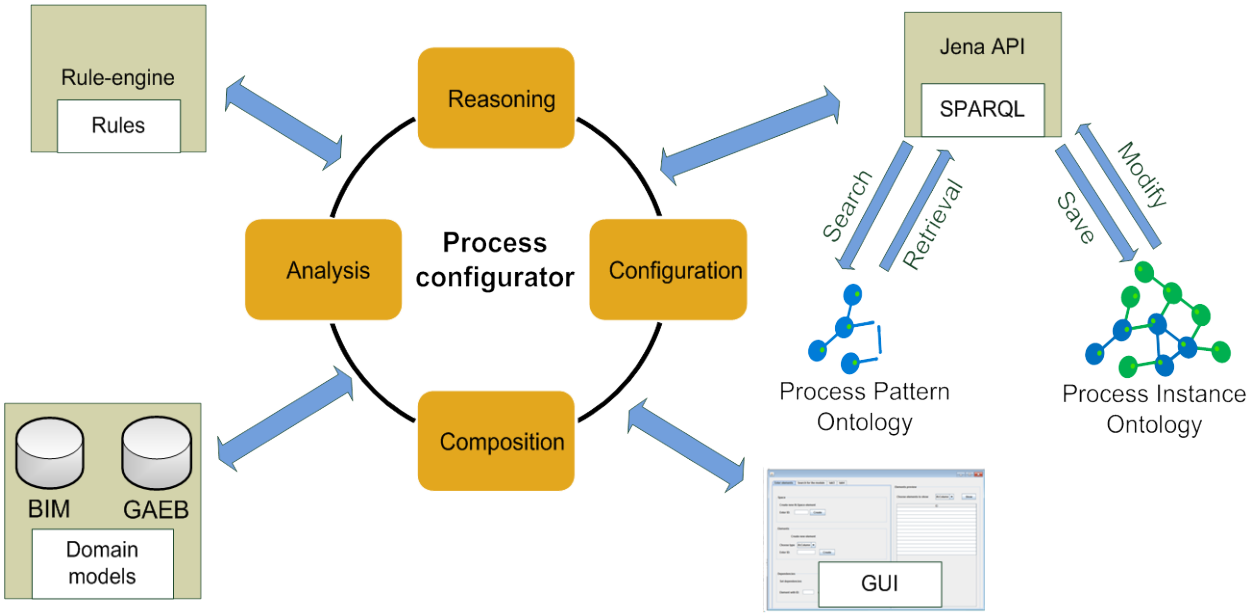


Figure 1: Principal system architecture

In order to interact with other domain models we have defined corresponding concepts in our ontologies. Moreover, for the case of the standard Building Information Model (BIM) we keep the structure of these concepts similar to the original on's defined in IFC (ISO PAS 16739) data model; therefore we have in our ontologies concepts like *IfcBuilding*, *IfcStorey*, *IfcMaterial*, *IfcColumn* and so on. By querying the concrete data models we can not only instantiate these concepts but also obtain various additional topological and structural dependencies, such us: "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.

### 3.1 Process Pattern Ontology

The Process Pattern Ontology plays a superior role in the developed framework. The two main upper level concepts in this ontology are *Process* and *Object* (Figure 2).

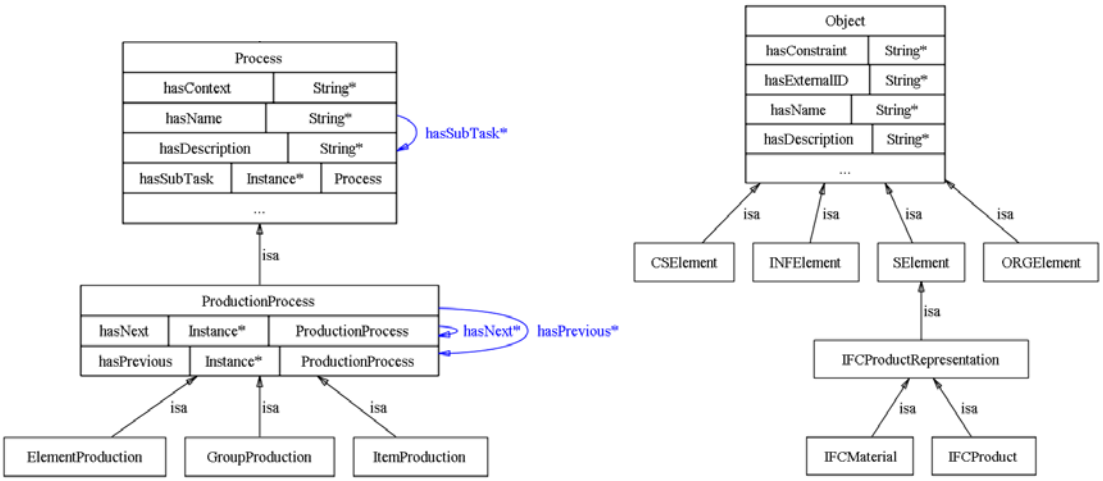


Figure 2: *Process* and *Object* concepts with their major properties and inter-relations

*Process* is a base concept to model all kinds of production processes that are considered in our work. In order to build a process hierarchy, an object property *hasSubTask* can be used. Other object properties are used to create a process workflow (*hasNext* and *hasPrevious*) or to reference a certain Object (*hasObject*).

Process Pattern Ontology stores typical process prototypes that can be used multiple times as predefined modules. Each pattern is an instantiated individual of some specific process concept that contains information about required resources and all related sub processes. An example of such a pattern that describes the process of column production with a crane is shown on Figure 3 below.

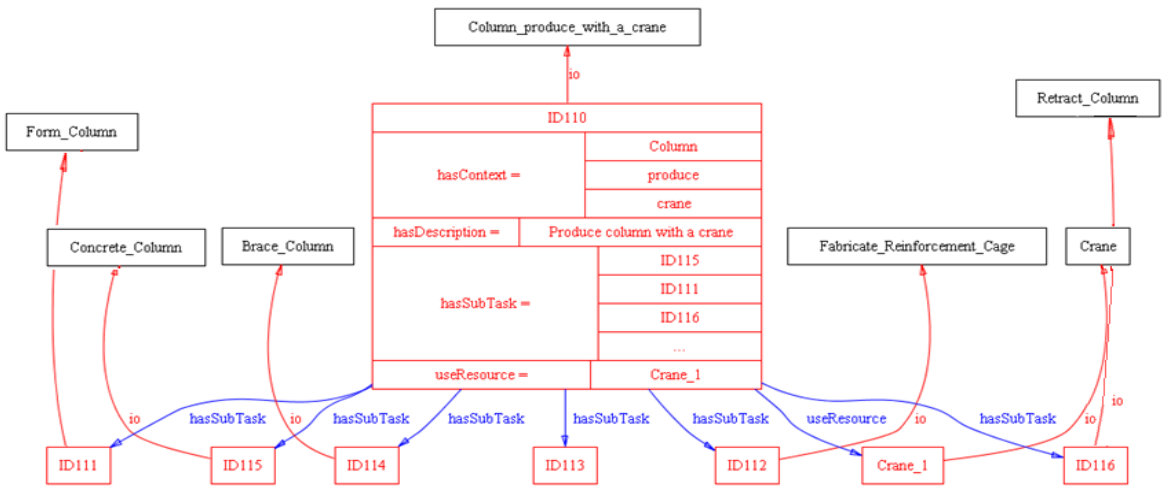


Figure 3: Pattern of the example process „Column produce with a crane“(presented partially)

Additionally, added meta-data information allows searching for the patterns and filtering them using certain conditions. Therefore we can search for the pattern that constructs some structural element using a specific building machine. An example for such a query is: "*Find all patterns that construct a column of some specific type with a crane*". As mentioned, to query our Process Pattern Ontology we use the SPARQL query language (W3C 2005). SPARQL support is available in the Jena framework and that allows us executing queries with the Jena API and integrating them in any Java application. Below is presented a simplified SPARQL query for the example mentioned above. This query uses four following variables:

- *Variable1* defines the class of the object ("column" or "wall"),
- *Variable2* defines the type of the object (for example "rounded column"),
- *Variable3* defines the class of the building machine ("crane" or "pump"),
- *Variable4* 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
       ?object mf:elementType    Variable2
       ?resource rdf:type    Variable3
      FILTER (REGEX(STR(?context),  Variable4))
    }
```

The Process Pattern Ontology is intended to be used for configuring many construction processes and therefore it is stored in the relational database running on a database server. This approach allows simultaneous access to the ontology by a large number of users and also increases access speed and robustness of the system.

### 3.2 Process Instance Ontology

In comparison with the Process Pattern Ontology, the content of the Process Instance Ontology is unique for each modelled 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 description of the process workflow (like *hasDuration* or *hasCost*). Initially this ontology has no instances, because they are added sequentially at runtime during the configuration process. The instantiation process of this ontology is semi-automatic, because of the fact that we use already predefined pattern structures from the Process Pattern Ontology, it is necessary to set concrete object identifiers in certain cases. Some identifiers could be generated automatically, however references to a specific construction object or building machine has to be assigned manually. 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 process description can be later exported from the ontology into a XML document that could be used as an input data for simulation process. Both ontologies are specified using the OWL (W3C 2004) language in order to facilitate their use in service-oriented architectures.

### 3.3 Rules

Due to the use of the ontological structure for the formalisation of construction process we have a possibility to apply a reasoning mechanism in order to check or modify the configured process. The use of rules allows us to set and then check some specific constructions constraints. Our first attempt was to use the Semantic Web Rule Language (SWRL), which is an expressive OWL-based rule language that allows writing rules that can be expressed in terms of OWL concepts and provides some basic reasoning support (W3C 2004b). However this reasoning support was not sufficient because SWRL provides monotonic inference only and therefore negation as failure and some other functionality are not supported. Moreover it is difficult to model advanced rule constructions and to introduce Java methods calls into SWRL rules and therefore the use of the general-purpose rule language was considered. For that purpose Drools, an open-sourced rules engine whose syntax supports more advanced rule constructs was chosen. Drools implements and extends the Rete algorithm (Forgy 1982) and is written and tailored for the Java language. The main technological challenge in using the general-purpose rule language is that an appropriate transformation from the ontological description into the rule-engine format is needed. Because of the fact that concepts from the ontology cannot be used directly for the facts base of the rule engine, the corresponding Java classes with appropriate set of properties were modelled. With the help of the Jena API these classes are instantiated with the corresponding individuals from the ontology. Moreover, only these ontological concepts, which are used in configuration rules, are transformed into the Java classes. The structure of the right hand side of the rule allows us to insert new facts in our working memory or execute some specific java method that could be used to write new information in the ontology. The suggested interaction between the rule engine and the Process Instance Ontology is presented on the Figure 4.

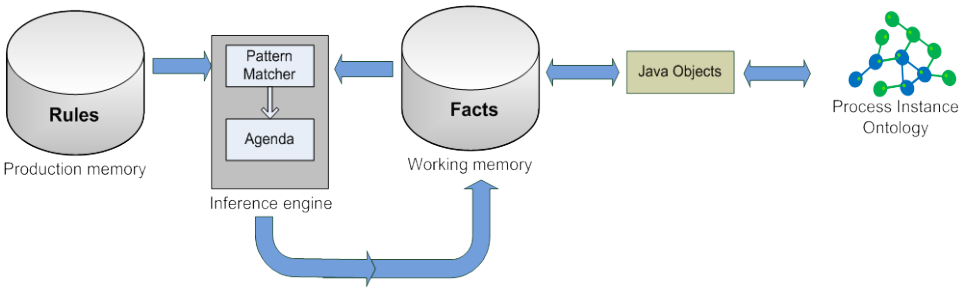


Figure 4: Interaction between the rule engine and the Process Instance Ontology

In our work rules are considered to be used for the two main purposes:

- Check specific construction constraints;
- Search for the possible process workflow and generate required relations.

One simple example of the rule that can check the consequence of the construction operations is presented below. The operation “Concrete a column” has to be performed after the operation “Form a column”, if they both operate with the same element:

$$hasNext(x,y) \rightarrow hasFollowing(x,y)$$

$$hasNext(x,y) \wedge hasNext(y,z) \rightarrow hasFollowing(x,z)$$

$$Concrete\_column(x) \wedge hasObject(x,z) \wedge Form\_column(y) \wedge hasObject(y,z) \wedge hasFollowing(x,y) \rightarrow System.output(\text{"Error Process:"} + y + \text{" must be performed before Process:"} + x)$$

Rules that generate process workflow are more complex and are themselves actually a set of rules. For example the rule “Construct walls before columns” provides few variants of the composition of the process submodules, where construction of the walls in one floor has to be performed before construction of the columns. This rule consists of a few subrules, where each of the subrule has its own priority for the execution.

## 4 Process Configurator and Use of the System

For the current prototype we do not consider all building elements and limit us just to the subdomain “structural concrete works”. The interaction between the ontologies and other components of the system is realized by a Java-based *Process Configurator*, which has the purpose to support top-down process planning whilst continuously considering respective building model data, as well as resources, equipment and overall strategic parameters. In the configuration process the user works with the graphical user interface of the Process Configurator (shown to the left on Figure 5). Ontology-based configuration of 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).

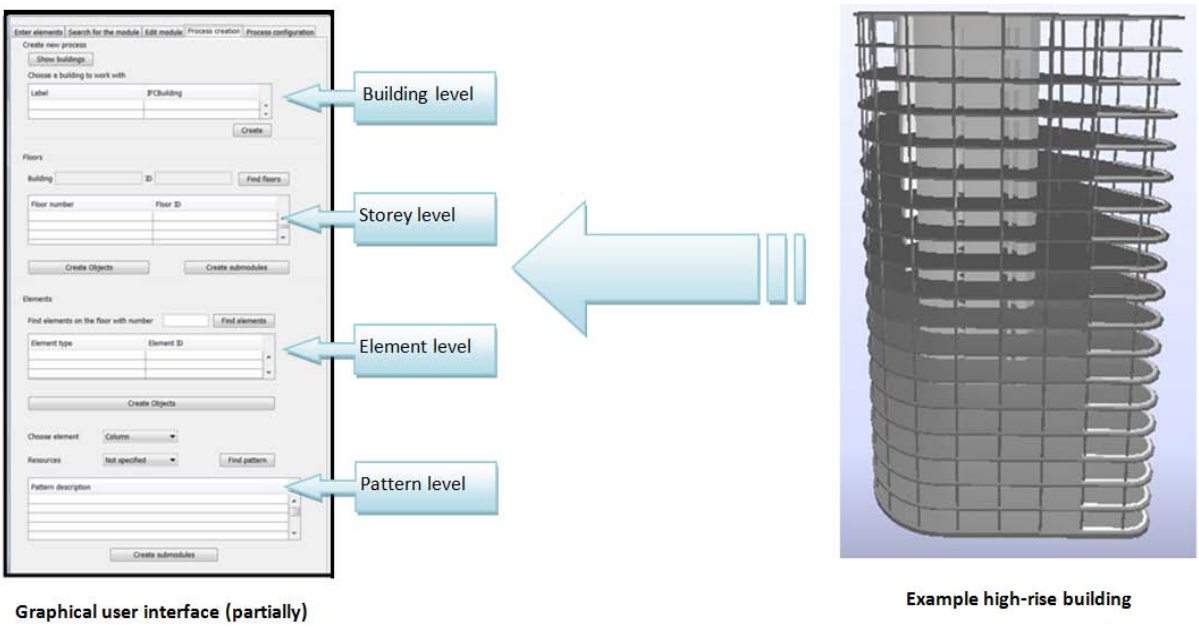


Figure 5: GUI of the Process Configurator and used building model

The entire configuration process is performed hierarchically. The user starts from the whole building and then details the process until it reaches building elements level. At the first step the user gets the required information from the Building Information Model and constructs general object and process models. These models contain only taxonomic information about the building and are used as basis for the further enrichments taking place at the following steps. For the BIM model standard IFC-based model data is used. For the example of a high-rise building we used in our studies (shown to the right on Figure 5), the object model includes information about all storeys in the building with some of their attributes and contained storey elements. The taxonomic relations between the elements are modelled with

the corresponding ontology relations (*contains* for the object model and *hasSubTask* for the process model). On the next step the user searches for the available pattern modules in the Process Pattern Ontology and then uses these patterns to configure a specific construction process that will be saved in the Process Instance Ontology. At this point the complete hierarchy of the process is modelled, however not all sequential relations (such as *hasNext* or *hasPrevious*) are represented. These missing relations can be set by the user manually or chosen from the variants proposed by the rule engine. An envisaged further development is integrating the developed prototype in the multi-model framework proposed in (Fuchs 2010). This will allow realising closer interaction with other domain models and improving usability of the system by providing some visualization components.

## 5 Conclusions

The described approach presents work in progress, performed in the frames of the German project Mefisto, partially founded by the German Ministry of Education and Research. To demonstrate and verify the proposed approach the model of a high-rise building was designed and formalised in IFC. A prototype Java-based application called Process Configurator is being implemented. Currently, it provides the defined ontology specifications, an initial knowledge base of process patterns, basic process rules and a set of construction process rules for the subdomain "structural concrete works". Initial studies showed that the proposed approach helps reducing the amount of work needed to design a construction process as well as improving its quality. However, further development efforts are needed regarding the definition of the new process patterns and extending the rule base.

## References


BAADER F., CALVANESE D., MCGUINESS D., NARDI D. & PATEL-SCHNEIDER P. F. /eds./ 2003. Description Logic Handbook: Theory, Implementations, and Applications, Cambridge Univ. Press, ISBN 978-0521876254.

BALI M.: Drools Jboss Rules 5.0 Developer's Guide, Packt Publishing Ltd. ISBN 978-1-847195-64-7, Birmingham, UK, 2009

FORGY C., "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", *Artificial Intelligence*, Volume 19 , 1982

FRIEDMAN-HILL E.: Jess in Action: Java Rule-Based Systems, Manning, ISBN 1-930110-89-8, Greenwich, CT, United States, 2003

FUCHS S., KATRANUSCHKOV P. & SCHERER R.J. (2010) : "A framework for multi-model collaboration and visualisation", *In: Proceedings of the ECPPM 2010, Cork, Ireland.*

GRIGORIS A., VAN HARMELEN F.: A Semantic Web Primer. The MIT Press, 2004.

KATRANUSCHKOV P., GEHRE A., SCHERER R. J.: *Reusable Process Patterns for Collaborative Work Environments in AEC*, In: ICE 2007 -Proceedings of the 13th International Conference on Concurrent Enterprising.,Centre of Concurrent Enterprise, Nottingham, UK, 2007

SCHERER R. J., KATRANUSCHKOV P. & RYBENKO K.: Description Logic Based Collaborative Process Management, Proc. 15th workshop of the European Group for Intelligent Computing in Engineering (EG-ICE), Plymouth, UK, 2008.

SCHERER R.J., SCHAPKE S.-E.& KATRANUSCHKOV P.(2010) "Concept of an Information Framework for Management, Simulation and Decision Making in Construction Projects", *In: Proceedings of the ECPPM 2010, Cork, Ireland.*

TAUSCHER E., MIKULAKOVA E., KÖNIG M., BEUCKE K.: Generating Construction Schedules with Case-Based Reasoning Support. In: Soibelman L., Akinci B. (Eds): American Society of Civil Engeneers (ASCE) Workshop 2007, Pittsburgh, July 2007.

W3C 2004 . OWL Web Ontology Language Reference. W3C Recommendation, 10.02.2004. Available at: http://www.w3.org/2004/OWL/

W3C 2004b. A Semantic Web Rule Language combining OWL and RuleML. W3C Recommendation, 21.05.2004. Available at: http://www.w3.org/Submission/SWRL/

W3C 2005 *SPARQL Query Language for RDF*. World Wide Web Consortium, February 2005, Available online at: http://www.w3.org/TR/2005/WD-rdf-sparql-query-20050217/

WU I., BORRMANN A., BEIßERT U., KÖNIG M., RANK E.: Bridge construction schedule generation with pattern-based construction methods and constraint-based simulation. Advanced Engineering Informatics 24(4): 379-388 (2010)