Ontology-Based Dynamic Process Support on the Example of Defect Management

P. Katranuschkov, K. Rybenko & R. J. Scherer
Institute of Construction Informatics, Technical University of Dresden, Germany

ABSTRACT: In construction projects, characterised by one-of-a-kind products and processes and frequently changing everyday tasks, achievement of dynamic process support is a critical success factor. However, current process management techniques rarely provide adequate dynamicity. In this paper we describe an ontology-based approach enabling dynamic construction of (sub)process chains with the help of pre-defined reusable business process objects coherently integrating processes with resources, services and responsible persons/teams for their execution. The paper presents the background of the approach, discusses the developed ontology framework and outlines the current environment and services with which it is used. The development work on domain-specific level targets the area of defect management which, due to the thousands of defects that have to be handled in parallel and the large number of unpredictable situations that have to be dealt with in a project, is an area of high process-related complexity. The reported research is largely done in the frames of the integrated German project BauVOGrid (2007-2010).

1 INTRODUCTION

Construction projects are one-of-a-kind undertakings characterised by one-of-a-kind products and processes, complex one-of-a-kind partner relationships, frequently changing tasks and high dependency on external factors like weather, transportation, various socio-political aspects etc. In such virtual organisation environments achievement of dynamic (ad hoc) process support is a critical issue as comprehensive process planning is rarely possible in advance. Indeed, modelling of the construction process can be done at the outset using e. g. the ARIS methodology (cf. Scheer 2000), and event-driven process chains (EPCs) in particular. However, EPCs lack fully formalised mechanisms that can enable flow control together with proper resource assignment to processes and explicitly defined actors/roles with respective process-related responsibilities, authorisation and access rights. Moreover, they provide only weak support to dynamic process modelling and management, basically requiring assembly of the model beforehand. To tackle these issues we suggest a formal holistic description of the overall model realised by a set of independent, yet inter-related ontologies. This provides an expedient mechanism for the achievement of interoperability in complex systems with multiple heterogeneous resources.

The developed layered set of inter-related ontologies (based on Description Logics) provides a clear distinction between concepts and instances as well as between generic and specific concepts in the maintained knowledge base. Consequently, knowledge representation of the process is done in three steps, from generic (meta) concepts, representing business process patterns (BPPs), through specific domain concepts to executable business process objects (BPOs). However, whilst the approach is designed to be highly generic and hence anticipated to be broadly applicable, the specific development carried out in the frames of the integrated German project BauVOGrid (cf. BauVOGrid 2009) focuses on the *defect management* process, also known as "snagging" in the UK. Appearing delusively simple, defect management is in fact a very complex process due to the thousands of defects that have to be handled in parallel, the constant (often controversial) inter-relationships of owner, general contractor and subcontractors, and the great number of ad hoc decisions to be taken. Therefore, it has been purposefully selected as a major scenario for the BauVOGrid platform which aims at achieving efficient VO cooperation and management in construction projects on the basis of distributed Grid and Web Services combined with semantic methods and goal-oriented process management.

The paper outlines the suggested overall approach, describes the developed ontology framework and provides an overview how it is currently used. More details are available in the public documents at http://www.bauvogrid.de.

2 BACKGROUND

In the following, the theoretical basis upon which we build our system is briefly outlined. This includes Description Logics and ontologies, event-driven process chains (EPC) and the suggested extension and modularisation of processes with the help of Business Process Objects (BPO).

2.1 *Description Logics*

Description logics (DL) is a knowledge representation formalism that can be used to represent the concept definitions of an application domain (known as terminological knowledge) in a structured and formally understood way (Baader et al. 2003). It refers also to the logic-based semantics which can be expressed by first-order predicate logic, a feature that was not available in its predecessors, frames and semantic networks. Today DL has become a corner stone of the Semantic Web for its use in the design and specification of ontologies via the Web Ontology language OWL (W3C 2004). OWL is also the language formalism used consistently in our work.

Elementary descriptions in DL are *atomic concepts* and *atomic roles* (also called *concept names* and *role names*). 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 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. The main constructs available in *ALCQI* are listed in Table 1 below.

Table 1. Syntax and semantics of description logics

Name	Syntax	Semantics
top concept	T	Δ^I
existentional 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	$C \cap D$	$C^I \cup D^I$
disjunction	$C \cup D$	$C^I \cap D^I$
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 $R \subseteq \Delta^I \times \Delta^I$. The inductive extension of the interpretation function to concept descriptions is also shown in Table 1.

A DL knowledge base usually consists of a set of terminological axioms (often called TBox) and a set of assertional axioms or assertions (often called ABox). An interpretation *I* is a model of a DL knowledge base if it is a model for the ABox and the TBox.

An equality whose left-hand side is an atomic concept is called *concept definition*. Axioms of the form $C \subseteq D$ for a complex description *C* are often called *general concept inclusion* axioms *(GCI)*. An interpretation *I satisfies* $C \subseteq D$ if $C^I \subseteq D^I$. Every concept definition $A \equiv C$ can be expressed using two GCIs: $A \subseteq C$ and $C \subseteq A$. Therefore a TBox can be seen as a finite set of GCIs. *I* is a *model* of a TBox *T* if it satisfies all GCIs in *T*.

An ABox assertion is of the form C(a), r(a,b), where a, b are individual names, C is a concept, and r a role name. An interpretation *I* additionally assigns to every individual name *a* an element $a^I \subseteq \Delta^I$. An interpretation *I satisfies C(a)* if $a^I \in C^I$ and *I satisfies r(a,b)* if $(a^I, b^I) \in R^I$. *I* is a *model* of an ABox *A* if it satisfies all assertions in *A*.

What makes description logics the formalism of choice is the fact that it defines a decidable fragment of first-order logic and, via OWL, a good background for a distributed modelling / service platform.

2.2 *Event-Driven Process Chains*

Event-Driven Process Chains (EPCs) are a business modelling technique that has become a de facto industry standard in the German-speaking countries, especially in conjunction with the ARIS methodology. Using certain normative extensions they integrate four major ERP modelling aspects, namely event, function, system / data and organisation (Figure 1).

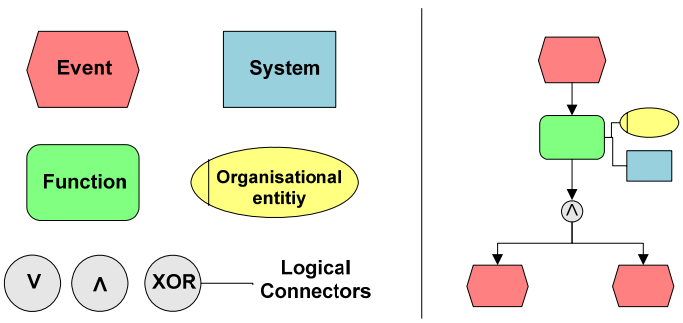

Figure 1. The basic EPC-elements (left) and a fraction of an EPC showing schematically their inter-relationships (right)

A *Function* can be understood as activity or action. Each function is preceded by a *before-event* and followed by an *after-event*. Syntactically, a function is a sentence, consisting of verb and noun, for example "Register defect", where the verb (or command)

transforms the noun (object). Functions can belong to one or more processes. Based on that observation, different classification criteria can be applied to functions. In general they can be grouped by objective, transformation, responsibility, or process, in which the functions are enrolled.

Events connect functions to provide a consistent workflow. They can be further categorised into *start-events*, *internal events* and *end-events*. From logical point of view, a two-valued *Status* has to be assigned to an event, indicating whether the event occurred or not.

Figure 2 shows an abstract process represented in DL. As in each EPC, it starts and ends with events. The top concepts are *Event* and *Function*, and it is easy to define *StartEvent*, *InternEvent* and *EndEvent* by means of DL. *StartEvent* is an event, after which some functions follow but there is no function that comes before this event, *InternEvent* has after- and before-functions, and *EndEvent* has a function that comes before the event but there is no function following after.

$$Function \subseteq \mathrm{T}$$
$$Event \subseteq \mathrm{T}$$
$$StartEvent \equiv Event \cap \exists hasAfterFunction.Function$$
$$\cap \neg \exists hasBeforeFunction.Function$$
$$InternalEvent \equiv Event \cap \exists hasAfterFunction.Function$$
$$\cap \exists hasBeforeFunction.Function$$
$$EndEvent \equiv Event \cap \neg \exists hasAfterFunction.Function$$
$$\cap \exists hasBeforeFunction.Function$$

Figure 2. Representation of functions and events in DL

Organisational Entity is used as anchor to the ARIS Organisational View describing organisational entities with their hierarchy and the communications between them. The concept of *roles* must also be defined here, to assign the rights for executing a concrete function.

Similarly, *System* is used as anchor to the ARIS Data View describing data objects, document and product data, as well as tools and services used to process this data in the context of the related function.

Output is the result of a process. It may comprise material and/or service results. The notion of output can be related to the notion of product.

The EPC itself is represented in the ARIS Control View in which the additional components *Flow Relation* and *Logical Connectors* are defined.

There exist also some syntactical rules for constructing an EPC, which must also be reflected in the corresponding ontology. On the basis of these observations a mapping strategy from EPC to DL was developed. It is presented briefly in Section 3.2.

2.3 *Business Process Objects*

A *Business Process Object* (BPO) can be understood as an extension of currently known Business Object specifications in that it enables better and more consistent binding of a real-world concept, representing a product or service which is the goal of a business activity, with the actual business process for the realisation of that activity (Katranuschkov et al. 2006; Keller 2007). A *Business Object* is typically comprised of a (sub) schema, population of the schema, methods assigned to the object providing various means to access and process the data, and (optionally) business rules providing quality management checks. A *Business Process Object* extends that definition by adding the process in which the business object is processed, the actor performing that process, the related actors to be notified and receiving results from the process, and the (set of) services and tools needed to perform the process. Hence, a BPO contains a network of objects representing a partial model, relations to distributed information resources and links to services/tools to process these resources. It consists essentially of the functions and their related resources belonging to a fragment of an EPC, grouped by specific criteria such as unique responsibility.

A BPO can be formally described as:

$$BPO\ (Name, F, E, SE, EE, C, R, O, Sim, Con) \quad (1)$$

where

F – finite set of functions
E – finite set of events
$SE \subseteq E$ – set of *StartEvents*
$EE \subseteq E$ – set of *EndEvents*
C – finite set of logical connectors
R – set of triples, representing the flow relations of the EPC, with $R \subseteq$ (*E, hasAfterFunction, F*), (*F, hasBeforeEvent, E*), (*E, hasBeforeFunction, F*), (*F, hasAfterEvent, E*), (*E, hasAfterConnector, C*), (*E, hasBeforeConnector, C*)
O – the organisational entity, responsible for executing the BPO
Sim – set of similar to a BPO other BPOs, having different *O*, *SE* or *EE*, as well as *Name*
Con – context of the BPO (important for defining and using various search criteria).

A BPO function f corresponds to the same EPC concept and can be defined as follows:

$$f\ (Name, A, Obj, R, S) \quad (2)$$

where

A – the action of the function
Obj – the object of the function
R – the resource(s) needed to execute the function
S – the system(s) or tool(s) required to perform the function.

According to the rules and conventions concerning EPCs, functions and events should be alternated, and regarding the connection of functions and events each function and each event may only have one input and one output connector. Therefore triples like (*F, hasAfterConnector, C*) can be omitted. Furthermore, the set of functions, connectors and events in a BPO have to be disjoint, i. e. $F \cap C \cap E = \varnothing$ and there should exist only one organisational entity for each BPO.

In summary, BPOs provide standard reusable process patterns, which can further serve for dynamic, IT-supported process configuration, instantiation and analyses on logical basis. The principal procedure is illustrated in the Figure 3 below.

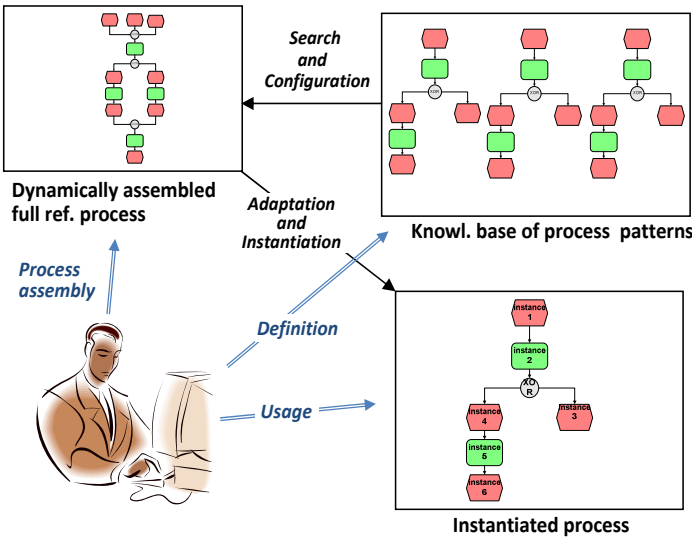

Figure 3. Principal use of BPOs stored in a DL knowledge base for the definition and instantiation of dynamic process chains

The first step is, starting from an idealised reference process model, to design the BPOs from which a specific EPC may be constructed (in our case for "Defect Management"). Secondly, having created a set of reusable BPO patterns (or shortly BPPs), an actual process flow can be dynamically assembled at execution time, pulling the required BPPs from the knowledge base and adapting them to the actual context. Adaptation rules applied for that purpose can be as simple as parameter variations but may also include elaborate procedures executed by some supporting tools. The latter, however, is dependent on engineering knowledge and might be a very complex task.

The criteria for the definition of reusable BPOs from an existing reference process model are identified as follows (Rybenko & Katranuschkov 2009):

Responsibility – An existing EPC should be divided into parts according to the defined responsibilities; for all functions in one BPO only one role is allowed to be responsible for the execution.

Modularity – A BPO should solve one specific problem; thus, it should represent one distinct process module.

Configurability – BPOs should be configurable; therefore they must be equipped with appropriate configuration rules and interface other BPOs only at event boundaries (and not at functions).

Time – In defining a BPO the anticipated time limits for its execution should be taken into account; even if the other criteria are met, the BPO should be divided in parts, if its execution time is inadequately long with regard to the overall project schedule (minimisation of time-dependent risks).

3 ONTOLOGY FRAMEWORK

In our approach, the DL knowledge base for collaborative process management is built in three distinct stages: (1) on generic level, (2) on domain-specific level, and (3) on run-time instance level (Scherer et al. 2008). This is done via a layered system of inter-related ontologies (Figure 4).

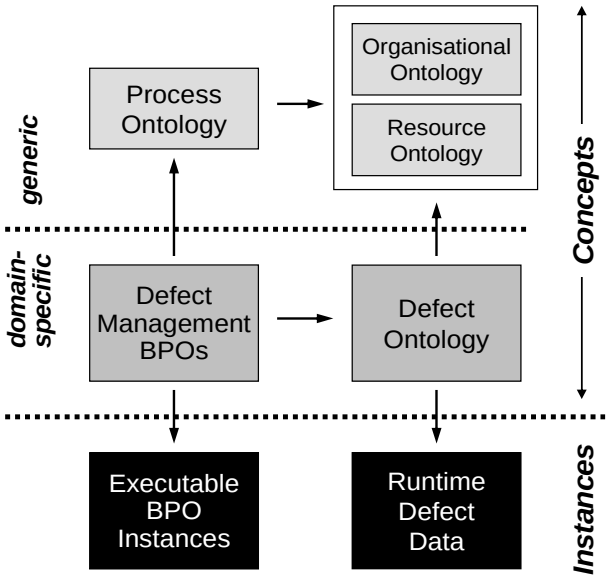

Figure 4. Schematic presentation of the developed ontology framework

At first, a generic widely reusable *Process Ontology* that describes the main features of EPCs and encompasses the definitions regarding the BPO concept is created. On that level, the generic *Resource* and *Organisational Ontologies* developed in the EU project InteliGrid (Dolenc et al. 2007) have been considered as well. The *Resource Ontology* is dedicated to the representation of all data resources available in a distributed project environment (files, documents, product models, product model views etc.), and the *Organisational Ontology* defines the concepts related to the structuring of a virtual project organisation (VO), i.e. the VO actors, persons, organisations and roles, together with the respective access control and authorisation constraints.

On the basis of these generic definitions, in the second stage the ontologies for a specific process type are created. In our case, this is the "Defect

Management" process. The *BPO Ontology* that specialises the BPO/EPC concepts and a *Defect Ontology* that describes the data for the defects itself via specialisation/extension of concepts from the *Resource* and *Organisational Ontologies* are defined here. In the *BPO Ontology* the idealised reference EPC for Defect Management is formalised as well.

The *Defect Ontology* contains the defect data of the involved project partners and provides a harmonised view on the distributed stored data. This enables purposeful and safe data handling by means of a set of implemented ontology-based services.

In the third stage, at run-time, the BPOs for a concrete process have to be instantiated and referenced with the run-time data corresponding to the data specification for a process. This means that the EPC for Defect Management will be instantiated multiple (up to several thousand) times for all actual defect cases in a specific construction project. The instances of these defects, i.e. the defect data, are maintained by the respective project partners according to their responsibilities and access rights but are inter-linked and referenced via the *Defect ontology*.

3.1 *Process Ontology*

The *Process Ontology* plays a leading role in the whole ontology framework. It contains such concepts as *BPO*, *EPC*, *Function*, *Event* and partially reflects the ARIS-methodology, thereby enabling the direct modelling (or mapping) of business processes in DL. It provides the description of an abstract EPC or an abstract fragment of an EPC (Business Process Pattern) and is also the basis for developing domain-specific BPO ontologies.

The main concepts of the *Process Ontology* are *Function* (Figure 5) and *BPO* (Figure 6).

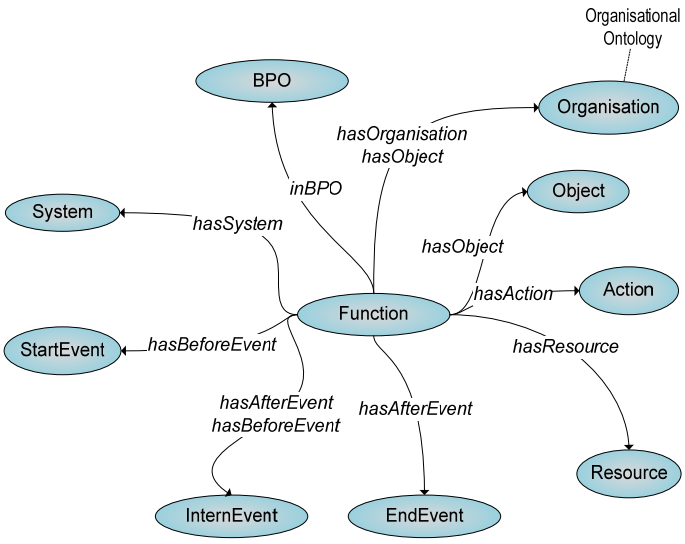

Figure 5. The concept *Function* of the *Process Ontology*

The relations assigned to the *Function* concept comprise *hasSystem, inBPO, hasOrganisation, hasObject, hasAction, hasResource*, as well as various flow relations like *hasBeforeFunction, hasAfterFunction, hasBeforeEvent, hasAfterEvent, hasBeforeConnector, hasAfterConnector* etc.

A *BPO* has also assigned *Resources*, start and end events (*StartsWith→Event*, *EndsWith→Event*), context data to facilitate search capabilities (*hasContext*) and, eventually, an EPC to which it has been attached (*inEPC→EPC*). Moreover, it contains a link to *Organisation* thereby identifying the person(s) responsible for its execution.

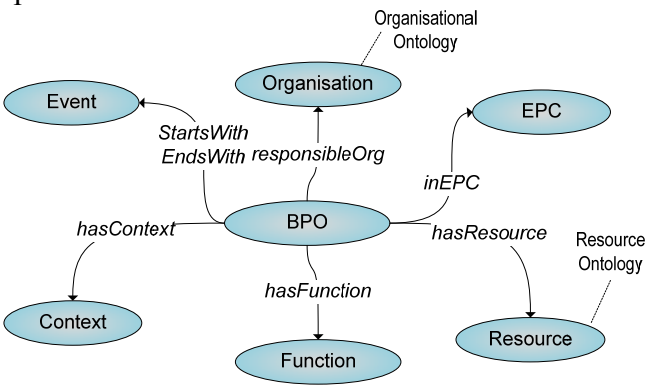

Figure 6. The concept *BPO* of the *Process Ontology*

A detailed example of a concrete EPC with concrete BPOs is given in Section 3.2 below.

3.2 *BPO Ontology for Defect Management*

As already mentioned, the purpose of a *BPO Ontology* is to represent a domain-specific target area, thereby subsuming as appropriate the high-level concepts of the Process Ontology. In our case this area is Defect Management. The main goal is to represent the reference EPC from which real project cases can be instantiated later by means of transformation and adaptation rules using modularised building blocks, the reusable BPOs.

As a first step in the ontology specification, the modularisation of the EPC is carried out applying the criteria listed in Section 2.3. In a second step, the actual formalisation of the BPOs in OWL is performed. This is illustrated below in DL-syntax on the example of the relatively simple BPO5.1 from the developed *BPO Ontology for Defect Management* (Figure 7).

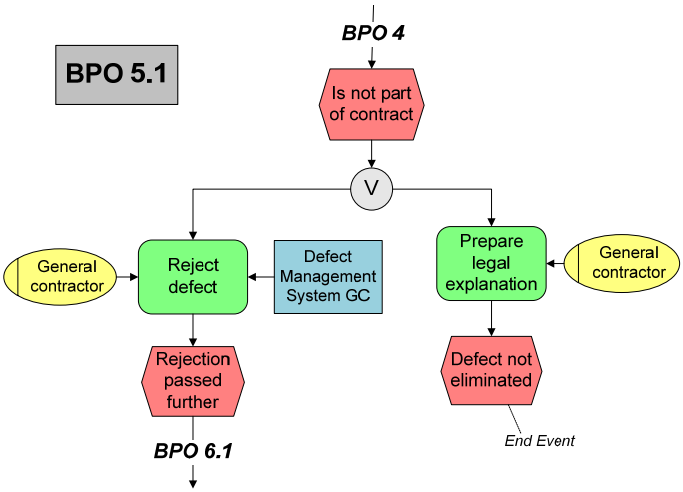

Figure 7. BPO5.1 „Defect processing if not part of contract"

The defined concepts for this example are:

Is_not_part_of_contract $\subseteq$ *InternEvent*
Rejection_passed_further $\subseteq$ *InternEvent*
Defect_not_eliminated $\subseteq$ *EndEvent*
Reject_defect $\subseteq$ *Function*
Prep_legal_explanation $\subseteq$ *Function*
General_Contractor $\subseteq$ *Organisation*
BPO5.1 $\subseteq$ *BPO*
DMS_GC : *System.*

The concept assertions are:

Is_not_part_of_contract (Is_not_part_of_contract)
Rejection_passed_further (Rejection_passed_further)
Defect_not_eliminated (Defect_not_eliminated)
Reject_defect (Reject_defect)
Prep_legal_explanation (Prep_legal_explanation)
General_Contractor (General_Contractor)
BPO5.1 (BPO5.1)
DMS_GC (DMS_GC)
EPC (EPC)
Context (Contract)
Action (prepare)
Action (reject)
Object (Legal_explanation)
Object (Defect).

Finally, the role assertions are defined as follows:

hasName (EPC, „Defect management EPC“)
hasName (BPO5.1,
„Defect procedure if not part of contract (GC)“)
hasName (Prep_legal_explanation, “54”)
hasName (Reject_defect, “24”)
hasEPC (BPO5.1, EPC)
hasSimilar (BPO5.1, BPO5.2)
hasOrganisation (BPO5.1, General_Contractor)
hasContext (BPO5.1, Contract)
StartsWithEvent (BPO5.1, Is_not_part_of_contract)
EndsWithEvent (BPO5.1, Rejection_passed_further)
EndsWithEvent (BPO5.1, Defect_not_eliminated)
hasBPO (Prep_legal_explanation, BPO5.1)
hasBPO (Reject_defect, BPO5.1)
hasAction (Prep_legal_explanation, prepare)
hasAction (Reject_defect, reject)
hasObject (Prep_legal_explanation, Legal_explanation)
hasObject (Reject_defect, Defect)
hasSystem (Reject_defect, DMS_GC)
hasAfterConnector (Is_not_part_of_contract, OR)
hasAfterFunction (Is_not_part_of_contract,
Prep_legal_explanation)
hasAfterFunction (Is_not_part_of_contract,
Reject_defect)
hasBeforeFunction (Rejection_passed_further,
Reject_defect)
hasBeforeFunction (Defect_not_eliminated,
Prep_legal_explanation).

The full set of the identified 11 BPOs modularising 57 individual reference process functions is shown in (Rybenko & Katranuschkov 2009).

The representation of the BPOs in DL provides all the necessary (meta) information to enable automated real-time instantiation. For example, the function “Reject_defect” can be instantiated with the individual “Reject_defect_33” that will be correspondingly represented in the ontology and inter-linked to all required additional data (responsible person, defect record, targeted receiver etc.) at run-time.

3.3 *Resource Ontology*

The *Resource Ontology*, originally developed in the InteliGrid-project (Gehre et al. 2007) and appropriately adopted here, targets the capturing of metadata describing various types of resources that are stored somewhere on a distributed project environment. It contains information *about* any kind of resource used in the environment but not the resource itself. However, capturing resource metadata does not only serve the purpose of establishing a central information service that holds URI references to distributed data. By annotating information resources with semantic metadata, software programs can automatically utilise the full context of what that information means and can make correct decisions about who may use the information and how. Also, in managing distributed digital resources the need for metadata that can support effective decision-making is even greater than with traditional information resources as there is less opportunity to recognise and understand a problem by merely looking at the target object. There is likely to be a much larger amount of information material to be managed (e.g. various multimedia data) so that management processes need to be automated as much as possible, based on easily interpreted rich metadata serving a broad range of requirements.

The main concept in the Resource Ontology is *Resource*. Figure 8 shows the subclass taxonomy below that core concept, including all explicitly modelled resources. As can be seen, the main distinction of resource types is between *ServiceResource* and *InformationResource*.

A further one-level specialisation of service resources is done by defining Storage Services and Processing Services. This classification is in line with the BPO definitions that need a clear distinction between the data provided by Storage Services and the processing functionality performed by Processing Services.

The *InformationResource* concept is specialised to *SingleInformationResource*, *ProductModel* and *ComplexInformationResource*.

SingleInformationResource is the main class for traditional singular resources like files and database entries, whereas *ComplexInformationResource* provides an abstract top-level class for a variety of composite resources that are commonly used in AEC projects. From the defined three subclasses of

SingleInformationResource it is anticipated that via *FileEntity* the majority of resources in a project will be captured. *FileEntity* is further subdivided to *StructuredFile* and *UnstructuredFile*, thereby distinguishing between files that can be parsed or analysed automatically by some third-party software, and files that provide non-structured data, as e.g. a scanned fax.

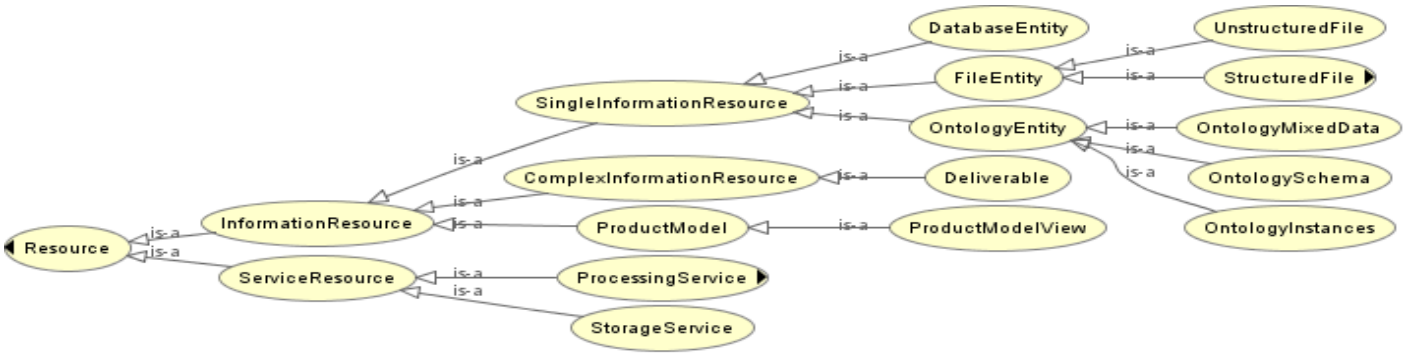

Figure 8. Taxonomy of the *Resource* concept

A further classification of unstructured files for organisational purposes is provided by means of a *ClassificationProfile* which defines a domain-specific classification mechanism that is also good enough for project specific classifications. Developed as ontology extension of the original Resource Ontology, instances of specific classification profiles can be dynamically assigned to resources meeting project specific requirements to the ordering of resources. This proves to be particularly useful in defect management for the classification of multi-modal media data associated to defects.

Single Information Resources can be pulled together in a *Complex Information Resource* – a simple collection mechanism that allows defining clusters of related resources. Complex Information Resources do not define process-like inter-relationships between the resources as defined in a Business Process Object but they can be very useful as integrated elements in a business process.

Another important concept of the Resource Ontology, not shown on Figure 8, is *ResourceProfile*. It provides a multi-facetted, wide ranging description of a resource that does not conform to any particular XML schema and has no particular single canonical or authoritative profile for a given resource. This approach is related to the idea of the semantic web with its highly distributed and heterogeneous pieces of information stored with free access to everybody and no guarantee of availability. However, for the purpose of metadata gathering in specific project environments such an irregular architecture will have serious problems with trust and quality of service. Hence, the really important feature of the Resource Profile concept is the treatment of the metadata of a resource as a *Learning Object* (Downes 2004) that gathers new information over time and from different "authors", serving different purposes, and described using heterogeneous facets of standards.

Finally, the concept of *AccessProfile* deserves to be mentioned. It enables separation of the resource description itself from the methods used to access the actual resource by different tools/systems. For example, one and the same file can have an FTP and a Web-DAV profile, thus providing different access methods to the information to the different parties.

3.4 *Organisational Ontology*

The *Organisational Ontology* combines in a coherent way various organisational aspects that are usually spread among different services, locations and information authorities, and are seldom represented explicitly. It enables storing the necessary management information into a single ontology storage, at the same time leaving sensitive partner data at place, guarded from unauthorised access. Its purpose is in the first place to enable proper authentication, authorisation and access control with regard to the management and the execution of BPOs, their related IT-services and the underlying resources. Accordingly, the Organisational Ontology is based on a couple of accepted standards from which it adopts essential concepts.

From the IFC standard developed by the IAI (IFC 2009), concept definitions are taken into account describing actor and project related information by contact details and metadata. Some of these concepts are shown on Figure 9 in their surrounding context.

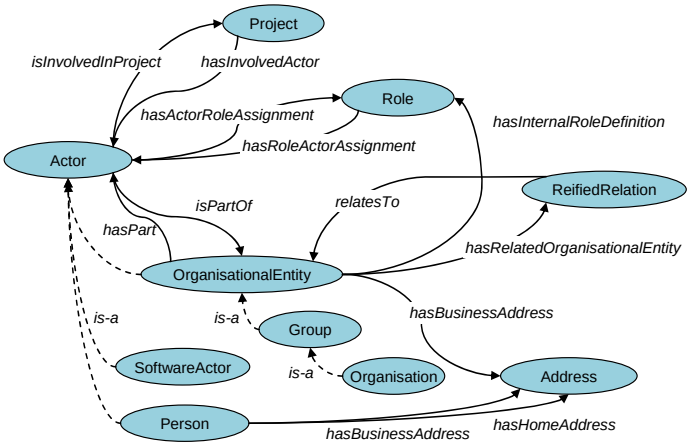

Figure 9. The *Actor* concept of the *Organisational Ontology* with its major inter-relationships

The adopted IFC definitions are in line with other standards describing these or similar concepts, such as CIM or GAEB. Therefore, information exchange with other systems should be possible without complex mappings.

Authorization aspects of the Organisational Ontology that are absent in IFC are modelled on the basis of adopted top-level concepts from the RBAC standard (Ferraiolo et al. 2001), as illustrated in Figure 10 below. The large amount of rules defined in RBAC is not modelled directly, since the Organisational Ontology is not used for dedicated management of access constraints. However, as not all information managed by an Ontology Service can be provided to each actor in a distributed environment, access information handled by an Authorization Service can be translated partially and stored in the RBAC related concepts of the Organisational Ontology. This information can then be used for restricting requests to any other ontology-based service. A difference to the original RBAC approach is the reuse of the *Actor* concept that replaces the simpler RBAC concept *User*. The Organisational Ontology has to serve a wider range of information requirements than RBAC; therefore the more sophisticated *Actor* concept outlined in Figure 9 needs to be applied. From Figure 10 it becomes clear that the *Role* concept is central to this approach. Permissions are granted to roles and actors can be dynamically linked to roles, with the additional possibility to define sessions in which a specific *Actor-Role* assignment is valid.

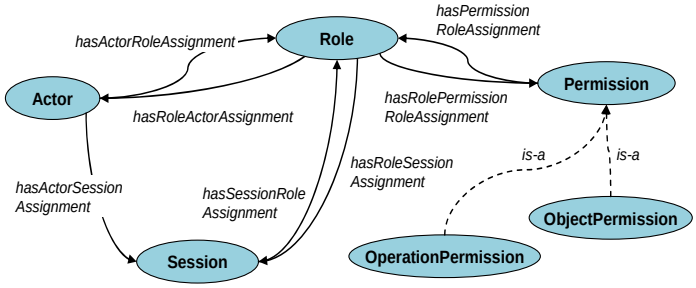

Figure 10. The principal concepts and inter-relationships in the Organisational Ontology providing RBAC support

Beside a number of data type properties, there are also object properties that relate *Actors* to *Roles* and *Projects*, as well as to the *OrganisationalEntity* aggregation concept and its subclasses, respectively.

3.5 *Defect Ontology*

Having constructed the Resource and Organisational Ontologies on the high level, development of a *Defect Ontology* enabling harmonised exchange and sharing of defect related data is a straight-forward task. The primary purpose of this ontology is to describe all necessary data regarding any project defect in such a way that distributed storage of the data and observation of public/private access to this data are warranted, whilst at the same time all common data can be coherently used by all affected parties in the defect management process. Therefore, beside specialisation of concepts of the Resource Ontology reflecting defect-specific information items such as *DefectType*, *ContractType*, *MediaData*, *ActiviyZone*, *Location* etc., concepts of the Organisational Ontology regarding the RBAC approach are adopted as well. These include *Partner* (as specialisation of *OrganisationalEntity*), *Role* etc. Here there are three principal actor roles involved: *Owner*, *General Contractor* (GC) and *Subcontractor* (SC). However, these roles can be easily further sub-classed if necessary. *Defect* itself is a typical *Complex Information Resource* as it may be associated to multi-modal information – database records and/or textual descriptions, drawings, photos, videos, audio recordings etc. (Figure 11).

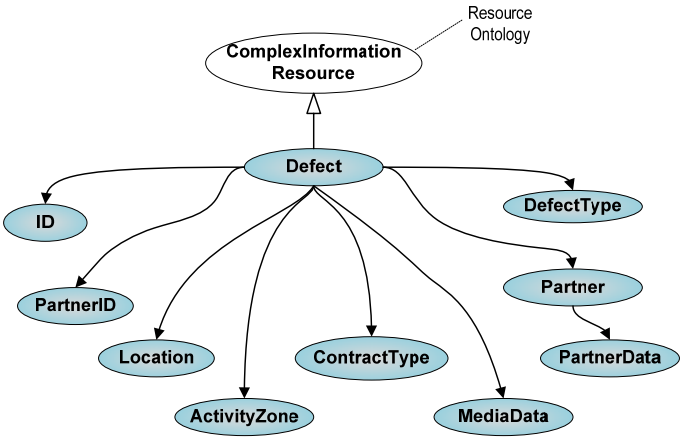

Figure 11. The *Defect* concept of the *Defect Ontology* with its major properties and inter-relationships

Exchange of defect data using the Defect Ontology is done via structured XML files or messages, based on a developed publicly available XML Schema specification published on the Internet (cf. http://www.bauvogrid.de/mangel/MangelSchema).

According to that schema, separate *Defect Records* or sets of such records can be exchanged or accessed on a project repository. The simplified example of a *Defect Record* provided below illustrates the outlined technique.

XML Defect Record Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<DefectRecord>
  <ID>81f7768d-199d-4b72-949e7</ID>
  <ActiveRoles>
    <Partner ID="ID_2" Role="SubContractor"/>
    <Partner ID="ID_1" Role="Contractor"/>
  </ActiveRoles>
  <Keywords>
    <User_defined> Wall cracks
    </User_defined>
  </Keywords>
  <Remark> Updated </Remark>
  <MediaData>IMAGE_00001.jpg</MediaData>
  <DefectType>Optical defect</DefectType>
  <DefectCause>unknown</DefectCause>
  ... ... ...
```

```
... ... ...
<Location>Berlin</Location>
<ContractType>Lump Sum</ContractType>
<Priority>1ow</Priority>
<Date>2009-02-13T09:30:47</Date>
<Deadline>2009-02-14</Deadline>
<Value Currency="EUR"> 199.99
</Value>
<Status>open</Status>
... ... ...
<PartnerRecord>
  <Partner IDREF="ID_2"/>
  <PartnerData>
    <Defect_NR>0815a</Defect_NR>
      <ResponsibleForDefect>
        SPINNLER GmbH
      </ResponsibleForDefect>
      <Informer>
        Euro-Trans GmbH & Co.
      </Informer>
      <Operator>
        Import Export GbR
      </Operator>
      <Controller>Mustermann</Controller>
      <StartDate>2009-02-13T09:30:47
      </StartDate>
      <FinishDate>2009-02-13T10:30:47
      </FinishDate>
      ... ... ...
  </PartnerData>
</PartnerRecord>
<PartnerRecord>
  <Partner IDREF="ID_1"/>
  <PartnerData>
  ... ... ...
  </PartnerData>
</PartnerRecord>
</DefectRecord>
```

As it can be seen, the role-based concept is used in the *Defect Record* in straight-forward manner via the `<PartnerRecord>` sections. All partners having to do with a certain defect are thereby specified with their active roles and are assigned their partner-specific data. However, as all the services providing access to *Defect Records* should observe the *Role-Actor* assignments made in a project, visible to each partner will be only the data partitions with respectively granted access rights. Hence, only as much as necessary information for each dynamically assigned defect management task will be made available. Private data stored locally can thereby be accessed via a 1:1 mapping of the defect's *ID* and the corresponding local *PartnerID*.

4 DEFECT MANAGEMENT SUPPORT USING THE DEVELOPED ONTOLOGIES

The described ontology framework for dynamic process support is freely accessible via the Internet at http://www.bauvogrid.de/ontologies/. It is available to any IT services, systems and tools that may use it on generic level, by providing their own domain-specific extensions, or, as in the case of the BauVOGrid project, specifically for the needs of Defect Management – the major targeted application scenario.

Indeed, defect management in any AEC project appears to be a rather complex process. Whilst the established idealised reference EPC for the treatment of one particular defect contains "only" 57 functions of which only some may have to be appropriately adapted and executed, there occur thousands of defects during construction that must be handled in parallel. According to industry experience, in a large project up to 50000 EPCs regarding defect and associated media data need to be managed, and 10000 to 15000 defects are rather typical. Such large numbers and the often highly complex contractor – subcontractor relationships emphasise the need of efficient and robust process-centred IT support.

In BauVOGrid, a number of Grid/Web Services and tools largely based on the described ontology framework are being developed to answer that need. These include:

- A set of *Basic Ontology Services* providing various support functions to the other, more application oriented services and tools
- A *Central Defect Management Service* and a *Central Media Data Management Service* that bring about the integration of existing, proprietary defect management systems via a range of query, processing, search and filter functions, thereby enabling owner – contractor – subcontractor cooperation
- *Mobile Services* for local positioning, RFID-based defect identification, barcode generation and automatic barcode recognition in ePhotos for dynamically assigning media and defect data
- A *Process Toolbox* for flexible, dynamic assembly of complex process chains on the basis of the ARIS methodology and the developed BPOs
- Further development of the *Grid Workflow Execution Service*, originally developed for the Fraunhofer Resource Grid and the EU project K-Wf Grid, for transferring business processes in executable workflows – see (Hoheisel 2008) and http://www.gridworkflow.org/kwfgrid/gwes/docs/.

At the time of this writing BauVOGrid is entering its final development phase. In a selected practice-relevant pilot project it will demonstrate how the developed services can be used to integrate three separate, proprietary defect management systems – in a secure VO network and in an efficient, process-oriented manner. It will also demonstrate how these systems can be expanded with further simulation and presentation tools. This is expected to highlight persuasive, new business perspectives for the construction industry and software providers.

5 CONCLUSIONS

In this paper, an approach for improved process management in construction on the basis of a well-defined system of ontologies was presented. To focus the research, to provide proof of concept and to achieve short-term practical exploitable results, the area of defect management has been addressed in all domain-specific considerations.

Formalisation of the defect management processes in ontologies and the use of these ontologies in a typical distributed IT environment with many heterogeneous resources appeared to have multiple advantages with regard to flexibility, interoperability and effective process management. Furthermore, by translating EPCs into Description Logics, introducing the concept of Business Process Objects and coupling these with resources, services and responsibilities, more robust and better formalised treatment of the ARIS methodology as well as dynamic process support, not available per se in ARIS, could be achieved.

Whilst more experimentation and real practice studies would definitely help to refine and improve the developed ontologies and their use in the area of defect management, an even more important future task is the application of the approach in other domain areas – to provide evidence of its wider applicability. This is intended in the large German integrated project MEFISTO inaugurated in mid 2009 that will be dealing with various complex management, controlling, simulation and decision-making tasks.

ACKNOWLEDGEMENTS

The concepts presented in this paper have been developed in the frames of the EU project InteliGrid, funded partially by the European Commission, and the German BauVOGrid project, funded partially by the German Ministry of Education and Research (BMBF). Their support and the support of the industry partners in these projects are herewith gratefully acknowledged.

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.

BauVOGrid 2009. *Project Description*, www.bauvogrid.de

Dolenc M., Katranuschkov P., Gehre A., Kurowski K. & Turk Z. 2007. The InteliGrid Platform for Virtual Organisations Interoperability, *ITcon,* Vol. 12/2007, pp. 459-477.

Downes S. 2004. Resource Profiles. *Journal of Interactive Media in Education*, 2004(5). Special Issue on the Educational Semantic Web. ISSN:1365-893X.

Ferraiolo D. F., Sandhu R., Gavrila S., Kuhn R. & Chandramouli R. 2001. Proposed NIST Standard for Role-Based Access Control, *ACM Transactions on Information and System Security* 4(3), pp. 227-274.

Gehre A., Katranuschkov P. & Scherer R. J. 2007. Managing Virtual Organisation Processes by Semantic Web Ontologies. In: Rebolj D. /ed./, Proc. CIB 24th W78 Conference Maribor "*Bringing ITC knowledge to work*", Univ. Library Maribor, ISBN 978-961-248-033-2, pp. 177-182.

Hoheisel A. 2008. Grid-Workflow-Management. In: Weisbecker A., Pfreundt F.-J., Linden J. & Unger S. /eds./ „*Fraunhofer Enterprise Grids – Software*", Fraunhofer IRB Verlag, Stuttgart, ISBN 978-3-8167-7804-2.

IFC 2009. *Industry Foundation Classes,* © International Alliance for Interoperability, http://www.iai-international.org

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, in: P. Cunningham and M. Cunningham /eds./ "*Exploiting the Knowledge Economy*", IOS Press, ISBN 1-58603-682-3, pp. 1055-1063.

Keller M. 2007. *Informationstechnisch unterstützte Kooperation bei Bauprojekten* (IT Supported Co-operation in Construction Projects; in German), Dissertation, Report No. 6, Institute of Construction Informatics, TU Dresden, 175 p.

Rybenko K. & Katranuschkov, P. 2009. BauVOGrid-Bericht A-3.1 *Ontologiespezifikation*. (BauVOGrid Report A-3.1 "Ontology Specification": in German), TU Dresden, 67 p.

Scherer R. J., Katranuschkov P. & Rybenko, K. 2008. Description Logic Based Collaborative Process Management. In: 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, 2008, ISBN 978-1-84102-191-1, pp. 291-302.

Sheer A.-W. 2000. *ARIS - Business Process Modeling*, Springer, ISBN 978-3540658351, 218 p.

W3C 2004. *OWL Web Ontology Language Reference*. W3C Recommendation, 10.02.04, http://www.w3.org/2004/OWL/

W3C 2004b. *XML Schema Second Edition*, Parts 0-2, W3C Recommendation, 28.10.2004, cf. http://www.w3.org/TR/