12

Use Case Semantics for Business Process Validation

Michael Wolters and German Nemirovskij
Albstadt-Sigmaringen University
Germany

1. Introduction

During the last decade the technologies which emerged in the context of Semantic Web research are developed to meet the challenges that are arising in the rapidly growing e-business domain. The main intention of such approaches is to relieve humans of decision making tasks requiring analysis and comparison of significant volumes of information (that is often heterogeneous or badly structured).
This chapter introduces an approach to quality management of business process models. It is focusing on the analysis of correspondences between process models and use cases that as we believe should be considered as operational expression of requirements to the business processes being modeled. The correspondences figured out in this procedure allow for assumptions to be made about the process models quality which is defined with respect to the requirement on business processes.
To facilitate the search for such correspondences an ontology describing both domains of interest, the one of process models and one of use case descriptions, was developed. The ontology when used as a common vocabulary facilitates homogeneous representation and efficient comparison of process models and use cases originally represented in different notations. This ontology together with the methodologies for converting process models and use case descriptions into the ontology based notation form the focal point of this chapter.
The quality aspects of business processes such as compatibility of collaborating sub processes or detection and avoiding of dead-locks in the process flow are addressed by modern tools for process modeling and management as well as by modern technologies such as Semantic Web Services. Yet the main question in the context of quality management remains the assessment of process validity.
The validity of a business process is basically its ability to tackle the use cases that are typically specified in the beginning of the business process development. At the same time we believe that the validation of business processes should be carried out with respect to their "life cycle" starting with the requirement definition, followed by business processes design and proceeding further with selection, development and orchestration of (web) services and finally ending with processing and testing. The earlier in the life cycle the validation takes place, the easier it is to change the process being developed, and the more

efficient is the development procedure. Therefore we think of the comparison of use cases and business process models as key approach to the validation of business processes.
On this purpose use cases and process models should at first be specified using the same notation. After this step their specifications must be mapped onto each other and compared semantically, e. g. with respect to the meaning of terms expressed by relations between them. Finally the inconsistencies or conflicts among them should be identified and assessed.
To carry out these three steps a framework dedicated to this task is required. Such a framework includes the following components:
- An ontology defining the common vocabulary and relations between terms that occur in use case descriptions and in process models. Though there isn't any formal standard for use case descriptions they usually fulfill a kind of latent standard and hence use similar vocabulary and structures. The vocabulary of process models on the other hand is restricted by the notation used for modeling, e. g. BPMN.
- An information-extraction mechanism for informally described and tabular structured use cases, whereby the information extraction is currently implemented using the semantic annotation approach. The result of this process is a machine-readable data structure referring to the common ontology mentioned above.
- A mechanism for transforming formally specified process models (e. g. with WS-BPEL) into a machine-readable data structure again referring to the same ontology.
- A tool for mapping and comparing this kind of data structures, detection of inconsistencies and conflicts and finally the graphic visualization of the comparison results.
In order not to go beyond the scope of this chapter we will concentrate on the first of the above mentioned components. The other will be part of our further research.

2. Business process specification

2.1 Models for business processes

Non- or semi-formally a business process is often described as a set of activities or tasks carried out by machines or humans and sequenced by a set of executive rules and constraints. However such a description does not make any difference between activities carried out in reality within very certain terms of time, e.g. *from 12:00 to 12:15 on 19.04.2009*, by concrete actors, e.g. by a clerk *Mr. Smith* or by an application *AXF#675* hosted on a computer with *IP 192.168.1.1*, and those activities that are specified in the form of tasks, directives or assignments for execution and used as a reference by concrete actors at a specific time.
Yet, in a formal view on business processes such a distinction is taken into account: The term *Business Process Model* is defined basically as a set of sequenced activity models or tasks. A *Business Process Instance* consists basically of activity or task instances (in this paper we use these terms as synonyms) and represents an application of a business process model by specific actors and under concrete circumstances and terms of time.
Following the formal definition, every business process model consists of nodes and directed edges. The latter expresses the relationship respectively the control flow between the different nodes within process models, whereas every edge has a conjunction with exactly two nodes and each node has at least one associated edge. According to (Weske, 2007 pp. 89) a node may describe an activity model, an event model or a gateway model.

Activity models represent the work units that have to be performed to fulfill the goal of the business process. They have always exactly one incoming and one outgoing edge.
Event Models stand for the occurrence of states relevant for the business process.
The control flow of activities, including sequences, split or join nodes, is expressed by *Gateway Models*.
Business process models can be specified using different notations. One of them is Event-driven Process Chains (EPC) (Tsai et al., 2006). EPC models are semi-formal and if used as specifications for automated execution often need additional explanations. Petri Net is an alternative notation that is more formal than EPC (Desel & Juhás, 2001). However Petri Net notation may be less expressive for human actors, especially if a large business process is specified. This special concept is described in greater detail in section 3.3.
In the last years however a comparatively new (developed 2002) Business Process Modeling Notation BPMN (Allweyer, 2008) became highly popular. The reason for this is that BPMN assembles a number of advantageous concepts known from preceding notations. One of the main advantages of BPMN is that it provides of a good comprehensibility for both business analysts who create processes and for technical developers who have to implement them.

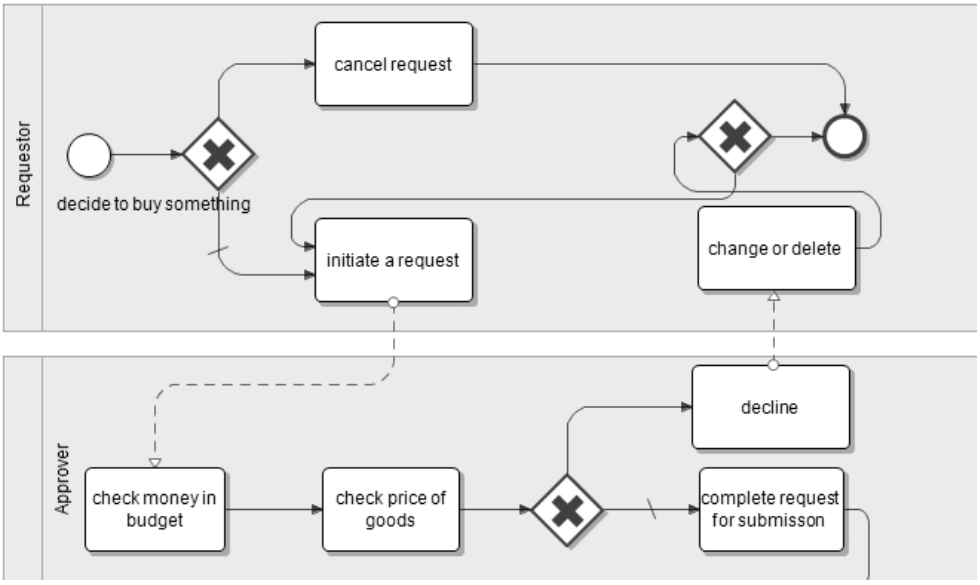

Fig. 1. Detail of a business process diagram created with Intalio Designer[1]

A sample business process model shown in figure 1[2] is specified by means of BPMN. It contains each of the three different node types mentioned above as well as the edges

[1] Intalio-Designer was one of the first implementations of BPMN, see http://www.intalio.com/products/designer/ for details.
[2] The business process model displayed in figure 1 is a simplified implementation of a use case developed by (Cockburn, 2001) using the BPMN. This specific use case example is described in greater detail in the chapter "Use cases".

represented by directed lines between the nodes. The two circles in the upper part of the picture specify event model nodes, whereas the left circle represents an initial event and the right (bold) circle represents a final event. Gateway models are displayed by diamonds, in this case used as XOR-splits, denoting the begin and the end of alternatively executed activity sequences. Finally the rectangles with rounded edges represent activity models. Another concept of BPMN are swimlanes modeling organizational aspects of a business process. Swimlanes can contain one pool (representing whole organizations) and two or more lanes (representing business entities like departments or single people). A pool can be seen in the upper part of figure 1 displayed as a grey colored rectangle labeled with "Requestor" at the left side of it. Lanes can split a pool in different parts by adding a horizontal line. The upper part of a lane called "Approver" is shown in figure 1 as well.

2.2 WS-BPEL

To enable automation of complex tasks such as processing, comparison or evaluation of business process models (the latter is strongly related to the approach described in this chapter) a formal executable specification of such models is needed. In this context Web Services Business Process Execution Language (WS-BPEL or short BPEL) developed basically for execution of web services carrying out single activities within a business process emerged as a de-facto standard. Furthermore there are a lot of BPM-tools and -IDEs that support an automatic BPEL generation. For example the Eclipse-based Intalio Designer (used to create figure 1) generates BPEL code out of BPMN based process models.
WS-BPEL was developed out of two other concepts: IBM's "Web Service Flow Language" and Microsoft's "XLANG". Thus BPEL adopted the XML based part from the Web Service Flow Language and the block structure part from XLANG. By now WS-BPEL has reached version 2.0 (published in May 2007) enhanced by the OASIS[3] group.
The different elements used in BPEL scripts can be divided into *basic activities* and *structured activities*. Some of the basic activities are: *Invoke*, *Receive*, *Reply*, *Assign* and *Empty*. *Sequence*, *Flow*, *While* and *Switch* are some structured activities. More information about WS-BPEL can be found in (OASIS, 2007). Figure 6 on page 9 shows an example of a BPEL code.

3. Informal and semi-formal scenario descriptions

3.1 Aims and goals for the application of scenarios

In contrast to the formal representation of process models aiming at their automated processing the scenarios' descriptions are kept usually informal. Often the general aim of scenario development is to formulate requirements for business processes being developed or to achieve a more thorough understanding of business processes by human actors. In the projects for establishing of new and reengineering of existing business process scenarios are usually described in the beginning phase, before specification of business process models.
(Beringer, 1997) defines scenarios as "… a sequence of interaction instances and/or state changes of objects. Interaction and state changes are also called actions." She writes furthermore of triggering scenarios with "trigger events" and defines the latter as an

[3] OASIS = Organization for the Advancement of Structured Information Standards

interaction or another event. In our approach we also use the term *trigger* to stress the causal or invoking role of events for actions or action sequences (see below).
In relation to business process models, scenarios represent one possible flow of activities or *operations* carried out by interacting actors. So all alternative flows are composed in different single scenarios. However usually scenarios reflect only the surface of interaction, and hence contain only the actions *seen* by interacting actors. Therefore some actions that take place in the *background* of interaction process may stay out of scenario, even though they are parts of a corresponding business process model.
For the business process model shown in figure 1, a quiet simple and short scenario can be specified as follows. Starting with the initial event (respectively *trigger)* labelled "decide to buy something" this scenario would continue with the *operation* "cancel request" and end after this step. An alternative more complex scenario for the same business process model would be: "initiate a request", "check money in budget", "check price of goods" and "complete request for submission".
In the following sections we briefly describe most popular notations for the specification of business scenarios.

3.2 Use Cases

Use Cases were developed for requirements engineering within the software development process. First mentioned by Ivar Jacobsen (Jacobson et al., 1994), Use Cases became quite popular especially in object-oriented Software Engineering. By using this concept an interaction with a (computer) system is defined from a user's point of view. Different steps of activities have to be processed to reach a certain goal which was originally specified by this user.

Use Case #	Use Case 5: Buy Something	
Primary Actor	Requestor	
Goal in Context	Requestor buys something through a system, gets it. Does not include paying for it.	
Preconditions	none	
Postconditions	*Minimal Guarantees*	Every order sent out has been approved by a valid authorizer. Order was tracked so that company can be billed only for valid goods received
	Success Guarantee	Requestor has goods, correct budget ready to be debited.
Trigger	Requestor decides to buy something	
Main Success Scenario	*Step*	*Action*
	1	Requestor: initate a request
	2	Approver: check money in budget, check price of goods, complete request for submission
	3	Buyer: ceck contents of storage, find the best vendor for goods
	4	Authorizer: validate Approver's signature
	5	Buyer: complete request for ordering, initiate PO with Vendor
	...	...
Extensions	*Step*	*Branching Action*
	...	...
	1b	At any time prior of receiving goods, Requestor can cancel request
	...	...
	2c	Approver decines: Send back to Requestor for change or deletion
	...	...
Secondary Actor	Vendor	

Fig. 2. Example of a use case description in tabular format

So the functional requirements of a system's behaviour can be specified in an objective and simple way without any loss of semantics. A more specific formalism about how Use Cases should be written and developed was given later by Alistair Cockburn (Cockburn, 2001). He differentiates between so called "casual" and "fully dressed" use cases. In the latter all possible requirements of a user-system-interaction are documented in detail. An example of a typical fully dressed use case description (this one is a simplified version of a use case originally created by (Cockburn, 2001)) is shown in extracts in figure 2.
Meanwhile the Object Management Group (OMG) has integrated the concept of Use Cases as a standard notation by developing a special UML diagram for their graphical modeling (see at e. g. OMG's http://www.uml.org). Although this diagram doesn't have the same detailed expressiveness as a fully dressed use case, UML can compensate this by an additional use of activity diagrams.
Text-based use case descriptions (usually documented in a tabular format) still lack such a widely approved formalism. However an analysis of text based use cases shows that such descriptions usually follow very few patterns determining a common vocabulary and structural principles. The set of such patterns can be generalized as a semi-formal *latent standard*.
This fact is acknowledged by a number of approaches for transformation of graphical and text based use case notation into each other (e. g. Pilarski & Knauss, 2008). In the approach described in this chapter we also utilize this fact for structure analysis and semantic annotation of text based use cases. Due to space limitation in this chapter we restrict the definition of use case description to text-based ones. Nevertheless the mapping of use cases and process models illustrated below can be applied as well for graphical use cases specified with UML.

3.3 Other Concepts for scenario description

Within this section we want to give a brief introduction to other notations for informal or semi-formal scenario descriptions. Even though these concepts differ from use cases in some aspects of representation they might be used for the validation of business process models in the way shown below.

Task Script

Ian Graham (Graham, 1995) created the concept of task scripts for modeling scenarios. A task script describes the interaction between actors and external entities with the objective of fulfilling the specified system's goal. In contrast to use cases, task scripts are implemented as objects so they can be organized into composition, classification, and usage structures. Among others Graham defines the following terms as parts of task scripts: Task Name, Task Body, Supertasks, Component Tasks, Task Attributes, Exceptions, Rules, etc. Task scripts are mainly used in the Financial Object-Oriented Rapid Analysis Method (FORAM) (Beringer, 1997).

Sequence diagram

The sequence diagram is part of the UML notation and a specialization of an interaction diagram. Sometimes sequence diagrams are called event-trace diagrams and are closely related to Message Sequence Charts (http://www.itu.int/rec/T-REC-Z.120). By using this kind of scenario description an interaction and an interchange of messages is graphically

defined. Each partner participating in an interaction is denoted by a vertical line called lifeline. Messages are illustrated by horizontal arrows in the order in which they occur. (Rupp et al., 2007)

Event diagram

Mathias Weske introduced in (Weske, 2007) a concept, called event diagram, which can be used for scenario description. In these event diagrams the time flow is shown in arrows going left to right whereas events are displayed by single bullets. Directed arcs represent the relationship of these events. For example of this scenario describing concept is shown in figure 3. Event diagrams may be used for representation of scenarios as well as process instances.

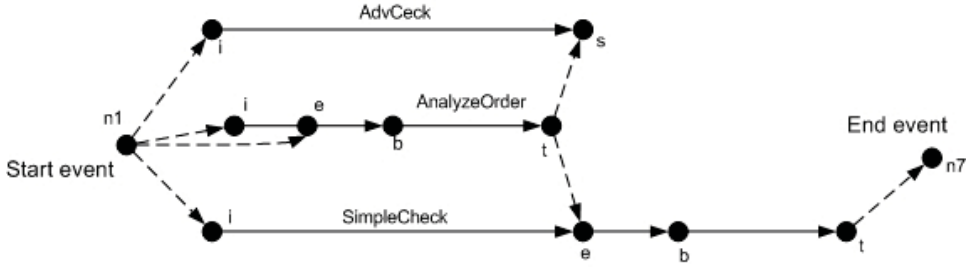

Fig. 3. Event Diagram example (see Weske, 2007, p. 95)

Petri net

Petri nets mentioned above can be used for specification of business process models as well as for representation of scenarios. The latter is especially realized by token concept, an approach for scenario-based modeling using Petri nets is introduced in (Fahland, 2008) for example.

Petri nets are represented as bipartite graphs and are composed of transitions, places and directed arcs which connect the places and transitions. Places can contain one or more token which can change their position when a transition *fires*. The current state of the petri net (thus the system or the process respectively that is modeled by this petri net) is represented by the position of its token(s).

In figure 4 an example of a simple Petri net is shown before (left side) and after (right side) a transition. While the token is at place p1 the transition t1 is enabled and may fire. After the transition the token was removed from p1 and added to p2.

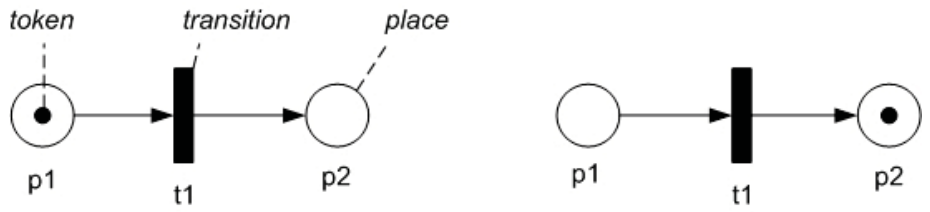

Fig. 4. Petri net example (Kashyap et al., 2008)

4. Validation of business process models using use case descriptions

As stated in the introduction the primary goal of the approach described in this paper is the validation of business process models. The validity of a business process model is assessed by searching for correspondences to the requirements for business processes and hence to the business scenarios, e. g. use case description, expressing these requirements.
Which correspondences are being searched for? When is a BPM valid? The formal definition of validity will be given in section 6. For now we define it informally.

Definition: A business process is valid with respect to a use case if the following rules are fulfilled:

1.) *Steps described in the use case correspond to particular activity model(s) of a considered business process*
2.) *The process flow connects these activity models to sub-processes corresponding to the Main Success Scenario and its Extensions specified in the use case*
3.) *In the business process model the connection of sub-processes corresponding to the Main Success Scenario and its Extensions is realized by means of gateway model(s) or event models*
4.) *The actor(s) is(are) involved in the use case correspond to the actor(s) involved in the business process being modeled*

As the space in this chapter is limited we restrict the definition to the four points introduced above. A definition of validity regarding other concepts that should be considered, for example trigger, is left out of our theory's demonstration but could be adopted analogous to our concept.
We will illustrate this definition by means of two examples. As we described in section 2 quite a few different notations can be used for specification of business process models. Figure 5 shows a sample of the mapping approach while BPMN is in use.

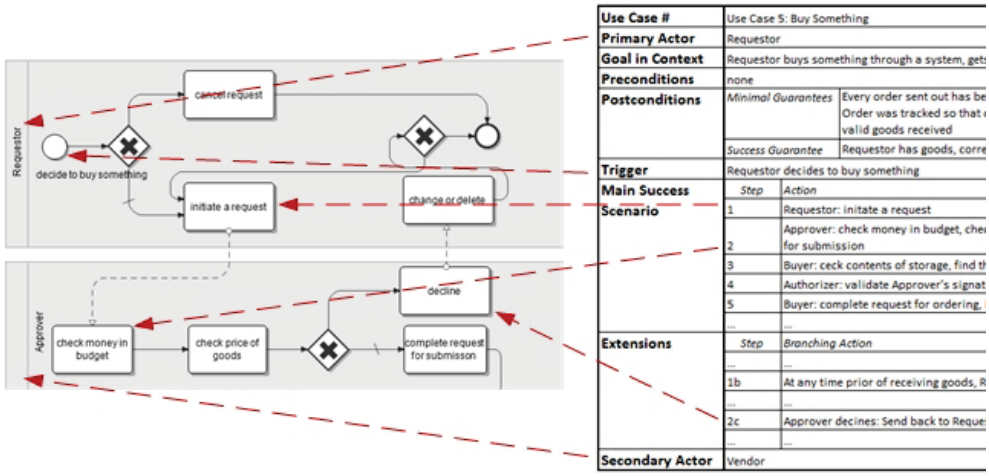

Fig. 5. Mapping use case onto business process model (BPMN)

1) As stated in the first rule of the definition above, all steps specified in the use case description (right part of the figure) are realized as activity models (rectangle with rounded edges). However, due to improvement of readability not all correspondences are shown in figure 5.
2) The starting point (*trigger*) of the use case equates to the start event model shown in the upper left part of figure 5 as a circle. The steps of the main success scenario (which are displayed in the use case specification *Scenario*) correspond to the activity models occurring in the process flow in the same sequence. This is true also for the steps of the *Extension-scenario*.
3) The steps specified in the use case as *Extension-scenario* correspond to the activity models of the sub-process started by a XOR-gateway (displayed as diamond).
4) The *primary* and *secondary actors* specified in the use case are mapped onto the pools of the business process model: The primary author *Requestor* is realized by the single pool in the upper part of the diagram and the secondary actor *Vendor* by another pool beneath it.

Figure 6 shows another mapping sample. However, a business process model is specified here using BPEL. Therefore the parts of the use case description have to be mapped to the BPEL components, e.g. the connection of *Main Success Scenario* and an *Extension* concept is realized by <*bpel:if*>- and <*bpel:else*>-constructs, while use case steps are represented by <*bpel:empty*>-, <*bpel:receive*>- or <*bpel:assign*>-tags.

Fig. 6. Mapping use case onto business process model (BPEL)

The examples shown above demonstrate that the mapping of business process models onto use cases can be applied to the effective validation of the former. People who have advanced knowledge in the field of process modeling can carry out this, doubtless complex and tedious work, in intuitive manner, even though business process models are often specified in different notations.

At the same time automation of this procedure promises to save significant resources while solving quality management tasks. Yet for machines searching for correspondences between use cases and business process models appears to be a big challenge. There are two reasons for that. First of all the comparison of artifacts specified in different notations (e.g. tabular use cases and graphical business process models) is difficult to formalize. Secondly use case specifications are informal per se.
However if searching for exact correspondences is replaced by searching for similarities, the automation of the mapping process becomes practicable. This idea is the foundation of the approach described in this paper.

5. Ontologies for Representation of Knowledge Semantics

5.1 Semantic Interoperability and Ontologies

For the automated search for correspondences between specifications of complex artifacts, e.g. between process models and use cases, a high level of semantic interoperability of these specifications is substantial.
In colloquial language semantic interoperability may be interpreted as a requirement, that terms or expressions occurring in different specification documents in equal form must have equal meaning. From the formal point of view the semantic interoperability of specifications facilitates avoiding of structural and semantic conflicts while interpreting. (Wache, 2003; Wache & Stuckenschmidt, 2001) describe basically three *structural* and two *semantic* conflicts.
To the category of *structural* conflicts belong:

- Bilateral conflicts, when in different description systems different identifiers, names or standard types (integer, float, string) for specification of the same world objects or artifacts are in use
- Multilateral Conflicts, when information represented in one source a single element can only partially be found in as a single element in another source.
- Meta-Level-Conflicts, when in different sources different modeling approaches are applied for representation of the same kind of information, e.g. a web site can be defined as a resource, as an entity or as an information unit

Semantic conflicts are:

- Data Conflicts, that appear when different metric systems or scales are used in different sources; however the single values located in these sources may look equal, in fact they should be distinguished with respect to the metric system currently in use, e.g. temperature according to Celsius or Fahrenheit scale.
- Domain Conflicts are situations when relations between classes specified in different classifications are not evident, e.g. overlapping: if a class of business tasks and a class of activity models specified in two classifications have a shared set of objects, however at the same time there are objects that belong to one class and do not belong to another.

To achieve the semantic interoperability of specifications, e. g. to avoid the conflicts listed above, the specifications' components, attributes and structure should be described using the same syntax and vocabulary. It is also necessary that within the domain of interest an agreement for the interpretation of expressions composed using the vocabulary is met. This

means that each term of the vocabulary should represent a group or a class of domain related objects sharing a well defined set of properties, whereby each of these properties is associated with a well defined set or space of values.
Such classes of objects are called concepts while the process of concept definition is called conceptualization. The explication of vocabulary shared by a group of specifications coming along with conceptualization of specific domain knowledge is commonly known as a Domain Ontology (compare to (Uschold & Gruninger 1996)). The descriptions of world objects or artifacts which are based on a domain ontology will be "understood" by actors (machines or humans) using the same ontology for the interpretation of the domain related expressions and specifications.

5.2 Foundation Ontologies

The development and usage of independent domain ontologies however reveals a number of drawbacks:

- Even within one specific domain the intention of ontology development is often a solution of a single specific task. From perspectives of different tasks however different conceptualization of the domain of interest may result. Therefore semantic interoperability of expressions and specifications based on different ontologies of the same domain is not guaranteed.
- Due to incompatibilities, e.g. conflicts described above, the knowledge exposed in different domain ontologies can't be shared easily and hence can't be taken into account for the solution of specific tasks. A good illustration for that are two ontologies developed in two strongly related projects aiming at the development of frameworks for semantic Business Process Management: *SemBiz* (http://www.sembiz.org) and *SUPER* (http://www.ip-super.org)[4]. Although the need for synergetic use of ontologies is stated on the SemBiz's web site, the lack of semantic interoperability between ontologies prevents the developers from achieving this goal.
- Extensions of domain ontologies to solve interdomain (mostly interdisciplinary) tasks may need a significant revision of given conceptualization. The reason for that is often the influence of the "other" domain, where the given conceptualization might be incorrect.
- If overlapping of different domains occurs the work for conceptualization of the overlapping parts may be carried out redundantly.

To address the problems listed above foundation ontologies (also known as upper-ontologies or top-level-ontologies) are developed. "Foundational ontologies are conceptualizations that contain specifications of domain independent concepts and relations based on formal principles derived from linguistics, philosophy, and mathematics." (Mika et al., 2004). In other words, development of a foundation ontology is an attempt to state a rough and abstract conceptualization of the world.
Domain ontologies usually use foundation ontologies as the "upper-level". Domain ontologies carry on the conceptualization started in a foundation ontology. Formally this

[4] Both ontologies are called BPMO (=Business Process Modeling Ontology)

means that in a domain ontology classes of objects or artifacts are defined as sub-classes of those specified in a foundation ontology.
For example the class "Business Process" in an ontology for the e-business domain can be specified as a sub-class of the class "Process" defined in a foundation ontology.
In spite of the fact that currently a number of foundation ontologies are known - e. g. *Cyc* one of the oldest ontology being developed since 1985 (http://www.cyc.com), *Basic Formal Ontology* that remarkably incorporates three-dimensional and four-dimensional (adding the time dimension) perspectives on reality (http://www.ifomis.org/bfo), or *Word Net* which exposes a set of psycholinguistic principles (http://wordnet.princeton.edu) - there is still no common standard or *the* foundation ontology that would be used as a base for all domain ontologies developed world wide. The question, if such an ontology will ever be developed is difficult to answer. There is a spectacular debate about feasibility and applicability of such common standard foundation ontology. The argumentation against such an ontology is based on the idea that ontologies developed by humans always expose the cultural, historical, linguistic and geographic context the developers live in. Hence the objective view on the world that could be reflected by a common standard foundation ontology cannot be stated by human beings. And vice versa: the feasibility of such an ontology comes close to the question if God exists.
Having in mind the conceptualization of Business Process Modeling domain on the one hand and use cases (UC) domain on the other hand we selected the Descriptive Ontology for Linguistic and Cognitive Engineering DOLCE (http://www.loa-cnr.it/DOLCE.html) as basis for our approach. DOLCE consists of different modules that can be used separately, e.g. *Plans*, *SpatialRelations*, *TemporalRelations*, *DOLCE-Lite* and *ExtendedDnS.* The latter module developed by Aldo Gangemi (Mika et al., 2004) is the *Description and Situation* ontology with an extended vocabulary for social reification. *DOLCE-Lite* which contains the most fundamental part of DOLCE and ExtendedDnS serve as base for all other modules.
DOLCE is one of the most popular foundation ontologies especially in the domain of e-Business that is related to Business Process Modeling and use cases as a super-domain to sub-domains. One of the basic ideas propagated in DOLCE is the distinction between perdurant and endurant objects. Perdurant objects such as *processes*, *events* or *activities live* in time whereas endurant objects are specifications related to perdurant objects separated from the time flow. *Workflows*, *plans*, *situations* or *people* belong to the endurant concepts (Magro & Goy, 2008).

5.3 ScIOn: Scenario and Business Process Model - Ontology

This paradigm of DOLCE corresponds well to the idea of business process specification. On the one hand business processes should be qualified as perdurant objects, e. g. artifacts *living* in time, artifacts that can be qualified by states achieved at certain moments described by certain temporal values. On the other hand the specification of such states related to the terms of time (that is a scenario) is endurant. The workflow specification (also known as a process model) is also endurant.
The description of relations between scenarios and process models as well as between their parts is the most important aim of the *ScIOn* (=*ScenarIo Ontology*) ontology developed by the authors of this paper to facilitate semantic interoperability of use cases (that is a particular form of scenario) and process model specifications.

The ScIOn that uses DOLCE as the upper-level ontology is shown in figure 7. All blue coloured oval forms show concepts of the DOLCE2.1-Lite-Plus. The fact that these concepts are specified in different sub-modules of DOLCE can be concluded from the name prefixes, e.g. concepts labelled with "edns:..." belong to the *ExtendedDnS* module. 13 concepts without prefixes belong to the core part of ScIOn. They all inherit properties of DOLCE concepts. The inheritance is represented by edges ending with triangle arrows. For example the core concept of ScIOn, *Scenario* is derived from the DOLCE concept *edns:path* that according to its description is "a concept used to sequence perdurant phenomena as components of some description". In turn the concept *Process Model* inherits the concept *sys:workflow* of the DOLCE module *Systems*.

ScIOn specifies a *Process Model*, according to definition from the section 2 as an aggregation (specified by diamond-ending arrows) of flow links (edges), gateways models, activity models (in ScIOn called *Operations*) and events models (ScIOn: *Triggers*). The process model elements are linked to each other using the DOLCE properties *edns:successor* and *edns:predecessor* and to the *Process Model* itself by the ScIOn property *elementOf*. At the same time *Operation* and *Trigger* are elements of the concept *Scenario*. In this case the relation is also implemented by the property *elementOf.*

By this means ScIOn facilitates direct mapping of process models (e. g. Business Process Models) onto *Scenarios*, and hence onto use cases. The concept Use Case, in turn, is specified as an aggregation of a set of scenarios, pre- and post-condition. By means of the DOLCE property *pla:main-goal* that was inherited from its super class *edns:plan* use case is associated with a certain goal, that should be reached after the use case is completed.

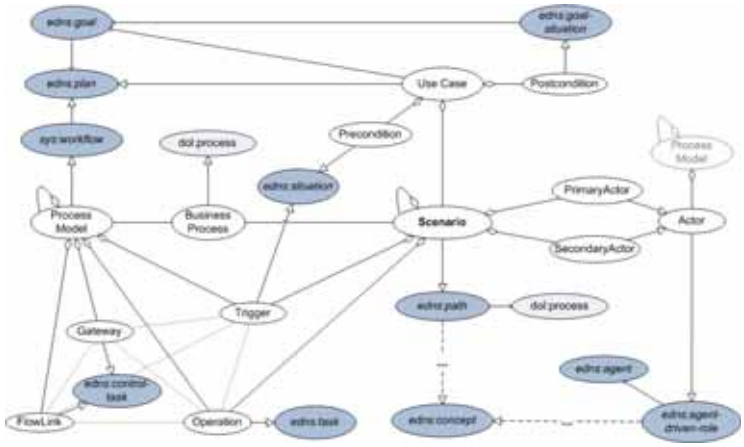

Fig. 7. Ontology for scenarios and business process models[5]

If one (perdurant) business process is on the one hand described by a process model and if there exists a set of scenarios federated in a use case, whose goal specifies one of the possible process results, we assume that the four rules of business process validity formulated in section 4 will be fulfilled.

[5] The grey colored concept "Process Model" above the concept "Actor" is the same as in the left part of figure 7 and was only added to not derange its clarity.

To demonstrate the contribution of ScIOn to semantic interoperability of use case and process model specifications we show a mapping of the use case and process model onto each other intermediated by ScIOn (compare to figure 5, showing the same mapping however without the ScIOn intermediation). For the simplification in figure 8 we show only 14 concepts: the 13 ScIOn core concepts and *edns:goal*, that is involved directly in the mapping. In this example we use a business process model specified with BPMN. As direct translation of BPMN to BPEL is proven to be feasible and even implemented in a number of tools such as *Intalio Designer* we skip the example showing the same process model specified in BPEL.

As shown in the example above ScIOn provides a common vocabulary for homogeneous representation of both artifacts being mapped onto each other. Furthermore, if expressed with ScIOn vocabulary the relations between artifacts' components and their semantics become explicitly and formally specified. The most important achievement of such representation is the prospect/opportunity for automation of correspondences identification, i. e. automation of validation process.

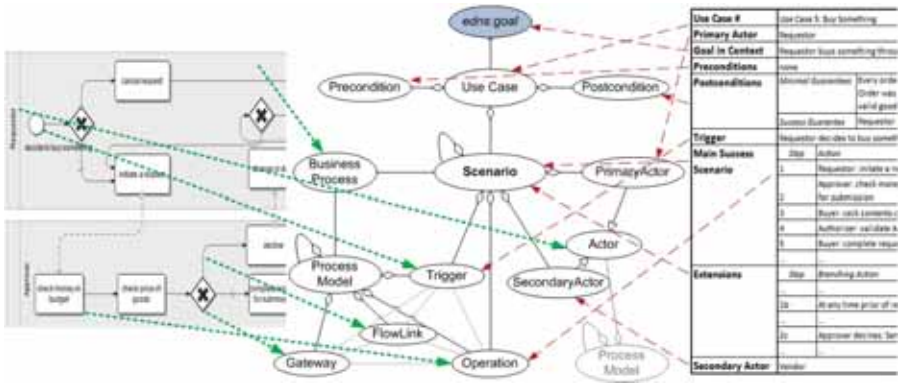

Fig. 8. Mapping use case to business process model (BPMN) intermediated by ScIOn

On the basis of ScIOn, business process models as well as use case descriptions can be semantically annotated or, in other terms, translated into *ScIOn language.* In this case their single parts, e. g. triggers and operations, are represented as instances of corresponding ScIOn concepts. Thus, machines or software programs respectively are now able to not only *understand* the meaning of these parts and of entire specifications but also to compare them effectively.

6. Automated validation of business processes using use case descriptions

6.1 Formal definition of business process model validity

To automate the validation of business process models the four validity rules defined in section 4 should be formulated in a manner that is understandable for machines, i.e. formally.

We will now formulate the four rules of business process models validity formally and with respect to the concepts definition given by ScIOn ontology:

An instance of the concept Process Model, *pm:ProcessModel* is valid with respect to an instance of the concept Use Case *uc:UseCase*,

$$pm \dashrightarrow uc$$

iff

1.) Each operation and trigger related to any scenario (that is related to the use case *uc* by the property *scenarioOf)* by the *elementOf* property, is related to the process model *pm* by the property *elementOf.*

$$(Trigger \sqcup Operation) \sqcap \exists elementOf.(Scenario \sqcap \exists scenarioOf.uc) \sqsubseteq \exists elementOf.pm \tag{1}$$

2.) The operations or triggers of scenarios participating in the use case *uc* keep their transitive *successor /predecessor* relations in the process model *pm.*

$$successor^{+} \sqsubseteq successor$$
$$predecessor^{+} \sqsubseteq predecessor$$

$$e\colon (Trigger \sqcup Operation) \sqcap \exists elementOf.(Scenario \sqcap \exists scenarioOf.uc)$$

$$\exists successor^{+}.e \sqcap \exists elementOf.(Scenario \sqcap \exists scenarioOf.uc) \sqsubseteq \exists successor^{+}.e \sqcap (Operation \sqcup Trigger \sqcup Gateway) \sqcap \exists elementOf.pm \tag{2}$$

$$\exists predecessor^{+}.e \sqcap \exists elementOf.(Scenario \sqcap \exists scenarioOf.uc) \sqsubseteq \exists predecessor^{+}.e \sqcap (Operation \sqcup Trigger \sqcup Gateway) \sqcap \exists elementOf.pm$$

3.) In the process model *pm* the connection of sub-processes corresponding to the *Scenarios* combined by the use case *uc* is realized by means of gateway model(s) or by event models. This means that the operations starting a scenario are preceded by a gateway or by a trigger in the process model *pm.*

$$Operation \sqcap \exists elementOf.(Scenario \sqcap \exists ScenarioOf.uc) \sqcap \nexists predecessor.(\exists elementOf.(Scenario \sqcap \exists ScenarioOf.uc)) \sqcap \exists predecessor(\exists elementOf.pm) \sqsubseteq (Trigger \sqcup Gateway) \tag{3}$$

4.) The actor(s) is(are) involved in the use case must be the actor(s) involved in the business process being modeled.

$$(Actor \sqcap \exists actorOf.(Scenario \sqcap \exists scenarioOf.uc)) \sqsubseteq (Actor \sqcap \exists actorOf.pm) \tag{4}$$

To sum the four rules described above form a base for the successful automated validation of business process models. These rules can be used as tasks for a semantic reasoner, a software that infers logical consequences from given knowledge, a set of facts or rules, that can carry out such validation, or be applied as axioms for specification of new process models.

6.2 Similarity and distance calculation

Yet, even if the analyzed instances don't match entirely they may have sufficient similarity to positively validate the business process under consideration. For example business processes can sometimes contain other business processes as intermediary but still fulfill the same function. Such a situation is illustrated by figure 9, where the middle activity of one process model is represented as sub-process in the other one. But both, the replaced activity and the more granular sub-process, realize the same task and thus both models are valid.
If the process model is changed as shown, a direct comparison with a use case could now result in a less than 100% correspondence. However such variation in composition of business processes needn't lead to negative validation results.

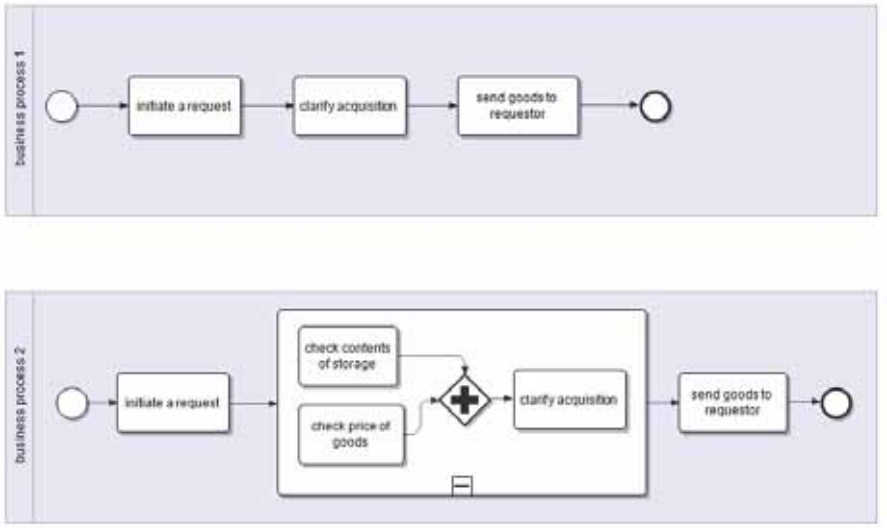

Fig. 9. Business process with sub-process

Consequently the new task arising in this context is to enrich the validation of process models by taking into account similarities between process models and uses cases.
Comparing two ontology individuals (instances or objects respectively) to analyze their similarity can be realized by a calculation of distance between this pair. An approach of such a distance calculation is introduced in (Maedche & Zacharias, 2002). According to this approach there are three dimensions of ontology-based similarity between objects: *taxonomy similarity (TS)*, *relation similarity (RS)* and *attribute similarity (AS)*.
One possible approach[6] to calculate similarity within TS is mentioned in (Lula & Paliwoda-Pękosz, 2008). There, similarity measures are based on path distance between concepts in the hierarchical tree of the ontology as applied in the following equation:

[6] There exist a couple of other approaches but describing all of them is out of the scope of this chapter.

$$sim_{WL}(C_1, C_2) = \frac{2 \times H}{N_1 + N_2 + 2 \times H} \quad (5)$$

N_1 and N_2 are the quantity of edges counted from the concepts C_1 and C_2 to the most specific common category C. H is the number of edges from this concept C to the root of the considered ontology.
We effectively use taxonomy similarity when comparing an individual of the ScIOn concept *Actor* with an individual of *PrimaryActor*. Although we deal with objects of two different classes a calculation of their taxonomy similarity gives us a measure for comparison of such objects.
In order to calculate similarity, the relation similarity RS dimension reflects the alikeness in relation to other objects. So when two compared instances are supposed to be similar to each other they should have relationships with concepts that are similar as well. For example if two compared *Operation* individuals have the same successors represented with the corresponding DOLCE property, they are considered to be similar Finally, using the *attribute similarity* dimension AS for comparison of instances of the same concept enables us for example to argue about similarity of operations by calculating distance between values of their *labels*. If labels of two operations are equal e.g. "initiate a request" on the figure above, there is high probability that we are dealing with two equal operations.
(Maedche & Zacharias, 2002) introduce the following formula for a calculation of similarity combining all three dimensions:

$$sim(I_i, I_j) = \frac{t \times TS(I_i, I_j) + r \times RS(I_i, I_j) + a \times AS(I_i, I_j)}{t + r + a} \quad (6)$$

In this formula *t, r,* and *a* stand for weights that can be specified separately to represent the potentially different importance of the similarity dimensions. I_i and I_j are the two instances or objects being compared.

7. Related Work

Quite a number of approaches deal with ontological representation of business process models. One of the most important of them is the EU funded research project called SUPER[7] (http://www.ip-super.org). It was aiming at the development of a framework for Semantic Business Process Management. The Web Service Modeling Ontology WSMO is another example. WSMO is used for *ontologization* of the BPM life cycle as well as concepts for semantic annotation of BPEL (called "sBPEL") and BPMN (called "sBPMN") are introduced (see for example in (Abramowicz, 2007)).
As mentioned in section 5.2 the Business Process Modeling Ontology (BPMO) was created as one of the deliverables of the SUPER project, to fulfill the specific requirements in this domain. Similar to the SUPER project a Business Process Modeling Ontology was developed

[7] SUPER = Semantics Utilised for Process Management within and between Enterprises

in the SemBiz workgroup. The intention of this project is the combination of the business level perspective and the technical implementation level in Business Process Management (BPM) using semantic descriptions of business processes.
To show parallels between the ScIOn approach and other projects in the related field on the one hand and to demonstrate the principle of ontology construction using upper-level ontologies on the other hand, we demonstrate two other options for ScIOn construction. The first of them (figure 10) exploits the *Business Process Modeling Ontology (BPMO)* developed in the SemBiz project.

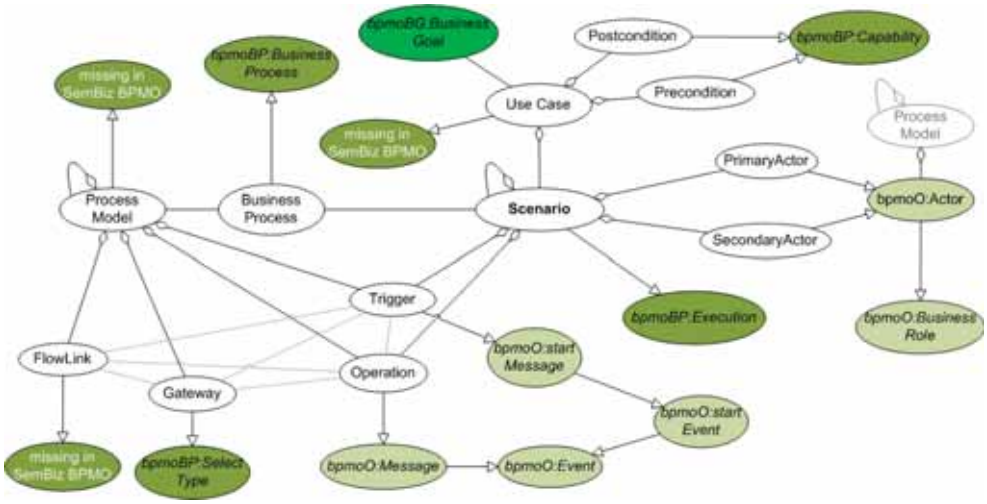

Fig. 10. Mapping to SemBiz BPMO

The second option demonstrates the use of SUPER BPMO in the role of an upper-level ontology of ScIOn:
Although these two ontologies were developed especially for business process modeling, we weren't able to find all required super-concepts for each of the 13 concepts of ScIOn. Furthermore the super-concepts we could identify aren't as suitable concerning semantical precision as the corresponding DOLCE counterparts. This fact demonstrates again the semantic expressiveness and role of foundation ontologies.
Therefore you can find some similarities between our approach and the one presented in (Mahl et al., 2007). The latter describes a bidirectional transformation between a DOLCE based reference ontology and BPEL. However, the approach is restricted to the support of cross-domain engineering in an e-business environment whereas the collaboration between different enterprises is improved by a common understanding supported by a specific ontology. The ontology-based approach enables a transformation of services and processes (developed in BPEL) into a standard representation so a cross-enterprise cooperation is possible. Another example of research aiming at semantic description of business processes is the one carried out at the Theory of Programming department8 of the Humboldt University in Berlin .

[8] See http://www2.informatik.hu-berlin.de/top/index.php?language=EN

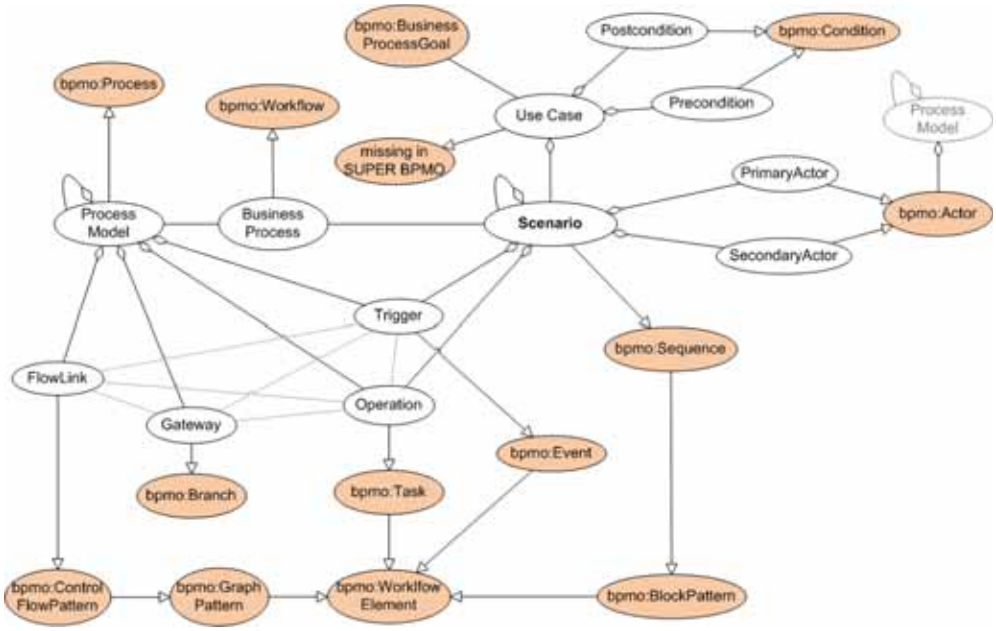

Fig. 11. Mapping to SUPER BPMO

It basically deals with problems of controllability in business process collaborations. One of the most interesting approaches generated in this context is introduced in (Lohmann et al., 2008). It focuses on using Petri nets semantics to describe business processes with the objective of transforming a BPEL into a Petri net model.

8. Conclusion and Future Work

This chapter described an approach addressing an important task in the context of quality management for business processes modelling. The approach is aiming at validating business process models with respect to the requirements on these models reflected in corresponding scenarios, e.g. use case descriptions. The approach focuses on homogeneous semantic-rich description of both issues (use cases and process models) followed by a comparison of the resulting specification documents.

Such a homogenisation is achieved by specification of elements (tasks, actions and events) used in the composition of business process models and use cases as individuals of concepts combined in one single ontology. In other words business process models specified using BPEL or BPMN and use cases basically written in text form are expressed using a common vocabulary. Therefore their direct comparison becomes a feasible task.

The ScIOn ontology described in this chapter is dedicated to this task. It gains high expressiveness from the DOLCE foundation ontology integrated by ScIOn as an upper level ontology. Application of ScIOn facilitates two methods of validity assessment for business process models. On the one hand a business process model is declared as completely valid with respect to a use case if both fulfil a set of axioms defined in ScIOn. On the other hand

ScIOn-based specification enables calculation of similarity between use cases and business process models and hence the relative validation of the business process models with respect to one particular use case.
Due to the lack of space this chapter does not cover the technology for automated translation of use case and process model specifications into the ScIOn vocabulary. Such technology is essential because the manual translation is highly complex and prone to errors. In this context the authors are currently working in two directions: 1) ScIOn based annotation of BPEL scripts and 2) ScIOn based information extraction from the tabular structured use case descriptions.
As stated in section 2 of this chapter there is a number of notations for specification of business process models. However during the last years the most popular notation became Business Process Model Notation (BPMN). A huge number of business process models developed recently was composed in BPMN. At the same time automated conversion of BPMN diagrams into executable WS-BPEL scripts containing commands for Web Service invocation is not a challenging task any more. It is widely applied and supported by a number of modelling tools such as Intalio Designer, Eclipse SOA Tools, Process Modeller for NS Visio, IBM Business Process Management (BPM) Suite, etc. Therefore, if technology for translation of BPEL scripts into ScIOn vocabulary was available, it would be sufficient for translation of a high number of business process models being developed in the close future.
Quite a different development can be observed concerning the notations for specification of use cases. Currently a number of various notations are in use: UML use case notation, tabular formatted text, activity diagrams, and some other notations (see section 3). To facilitate highly efficient validation of business process models a number of converting technologies should be developed: one for each existing notation. However tabular formatted text appears to be the simplest and hence the most popular form for the use cases' composition. Therefore the processing of use cases specified in this notation will be a future task of the ScIOn project.
Text structure analysis research is an important field in the information extraction technology. To the central works in this area belong among others (Tengli et al., 2004), (Tijerino et al., 2005) and (Gatterbauer et al., 2007). The technology dedicated to the special problem of structure analysis for use case specifications will incorporate some techniques described in these works. However, in contrast to the domain independent approaches presented in the papers mentioned above, structure analysis for use case specifications should be rather classified as a domain dependent one. Therefore it may rely on some document features, e. g. particular structural elements, typical for the domain of interest. This consideration let us suppose that the task to be solved is significantly simpler than a generic approach.

9. References

Abramowicz, W.; Filipowska, A.; Kaczmarek, M. & Kaczmarek, T. (2007). Semantically enhanced Business Process Modelling Notation, *Proceedings of the Workshop SBPM 2007*, ISSN 1613-0073, Innsbruck, April 2007, CEUR-WS, Aachen

Allweyer, T. (2008). *BPMN - Business Process Modeling Notation - Einführung in den Standard für die Geschäftsprozessmodellierung*, Books on Demand, ISBN 978-3-8370-7004-0, Norderstedt

Beringer, D. (1997). *Modelling global behaviour with scenarios in object-oriented analysis*, Ph. D. Thesis, Ecole Polytechnique Fédérale de Lausanne (EPFL), School of Computer and Communication Sciences, Lausanne, URL: http://library.epfl.ch/theses/?nr=1655, accessed 02/22/09

Cockburn, A. (2001). *Writing effective use cases*, Addison-Wesley, ISBN 0-201-70225-8, Boston

Desel, J. & Juhás, G. (2001). What Is a Petri Net? Informal Answers for the Informed Reader. *Lecture Notes in Computer Science*, Vol. 2128, January 2001, pp. 1-25, ISSN 0302-9743

Fahland, D. (2008). Oclets -- Scenario-Based Modeling with Petri Nets, *Proceedings of the 15th German Workshop on Algorithms and Tools for Petri Nets, AWPN 2008*, pp. 1-6, ISSN 1613-0073, Rostock, September 2008, CEUR-WS, Aachen

Graham, I. (1995). *Migrating to Object Technology*, Addison-Wesley, ISBN 0-201-59389-0, Reading

Gatterbauer, W. et al. (2007). Towards domain-independent information extraction from web tables, *Proceedings of the 16th international conference on World Wide Web*, pp. 71-80, ISBN 978-1-59593-654-7, Banff, May 2007, ACM, New York

Jacobson, I. et al. (1994). *Object-Oriented Software Engineering - A Use Case Driven Approach*, Addison-Wesley, ISBN 0-201-54435-0, Wokingham

Kashyap V.; Bussler, C. & Moran, M. (2008). *The Semantic Web - Semantics for Data and Services on the Web*, Springer, ISBN 978-3-540-76451-9, Berlin & Heidelberg

Lula, P. & Paliwoda-Pękosz G. (2008). An ontology-based cluster analysis framework, *Proceedings of the First International Workshop on Ontology-supported Business Intelligence, OBI 2008*, pp. 33-38, ISBN 978-1-60558-219-1, Karlsruhe, October 2008, ACM Press, New York

Lohmann, N.; Verbeek, E.; Ouyang, C. & Stahl, C. (2008). Comparing and Evaluating Petri Net Semantics for BPEL. *International Journal of Business Process Integration and Management (IJBPIM)*, (Accepted for publication), ISSN 1741-8763, URL: http://www2.informatik.hu-berlin.de/top/download/publications/Lohmann VOS2008_ ijbpim.pdf, accessed 04/20/09

Maedche, A. & Zacharias, V. (2002). Clustering Ontology-Based Metadata in the Semantic Web. *Lecture Notes in Computer Science*, Vol. 2431, January 2002, pp. 383-408, ISSN 0302-9743

Magro, D. & Goy, A. (2008). The Business Knowledge for Customer Relationship Management: an Ontological Perspective, *Proceedings of the First International Workshop on Ontology-supported Business Intelligence, OBI 2008*, pp. 33-38, ISBN 978-1-60558-219-1, Karlsruhe, October 2008, ACM Press, New York

Mahl, A.; Semenenko, A. & Ovtcharova (2007). Virtual organisation in cross domain engineering. *IFIP International Federation for Information Processing*, Vol. 243, September 2007, pp. 601-608, ISSN 1571-5736

Mika, P.; Sabou, M.; Gangemi, A. & Oberle, D. (2004). Foundations for OWL-S: Aligning OWL-S to DOLCE. *2004 AAAI Spring Symposium - Semantic Web Services*, No. SS-04-06, pp. 52-60, ISBN 1-57735-198-3

OASIS (2007). *Web Services Business Process Execution Language Version 2.0 - Primer*, OASIS, URL: http://docs.oasis-open.org/wsbpel/2.0/Primer/wsbpel-v2.0-Primer.pdf, accessed 04/13/09

OMG (2009). *Business Process Modeling Notation (BPMN) - Version 1.2*, Object Management Group, URL: http://www.omg.org/spec/BPMN/1.2/, accessed 02/23/09

Pilarski, J. & Knauss, E. (2008). Transformationen zwischen UML-Use-Case-Diagrammen und tabellarischen Darstellungen, *Proceedings of the Workshop on Domain-Specific Modeling Languages (DSML-2008)*, pp. 45-58, ISSN 1613-0073, Berlin, March 2008, CEUR-WS, Aachen

Rupp, C.; Queins, S. & Zengler. B. (2007). *UML 2 glasklar*, Hanser, ISBN 978-3-446-41118-0, München & Wien

Tengli, A. et al. (2004). Learning table extraction from examples, *Proceedings of the 20th international conference on Computational Linguistics*, pp. 987-993, Geneva, August 2004, ACM, New York

Tijerino, Y.A. et al. (2005). Towards Ontology Generation from Tables. *World Wide Web Journal*, Vol. 8, No. 3, September 2005, pp.261-285, ISSN:1386-145X

Tsai, A. et al. (2006). EPC Workflow Model to WIFA Model Conversion, *Proceedings of 2006 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2758-2763, ISBN 1-4244-0099-6, Taipei, October 2006, IEEE Xplore

Uschold, M. & Gruninger, M (1996). Ontologies: Principles, Methods and applications. *Knowledge Engineering Review*, Vol. 11, No. 2, 1996, pp. 93-155, ISSN 0269-8889

Wache, H. & Stuckenschmidt, H. (2001). Practical Context Transformation for Information System Interoperability. *Lecture Notes in Computer Science*, Vol. 2116, January 2001, pp. 367-380, ISSN 0302-9743

Wache, H. (2003). *Semantische Mediation für heterogene Informationsquellen* (Semantic mediation for heterogeneous information sources), Ph.D. Dissertation, Akademische Verlagsgesellschaft, ISBN 978-3-89838-261-8, Berlin

Weske, M. (2007). *Business Process Management - Concepts, Languages, Architectures*, Springer, ISBN 978-3-540-73521-2, Berlin & Heidelberg