

Modelling Languages and Semantic Integration in Business and IT worlds

Brayan Zimmerli, Patrick Moser

Business Information System, University of Northwestern Switzerland
{brayan.zimmerli, patrick.moser}@students.fhnw.ch
December 2008 – January 2009

Abstract: Businesses have processes, which need data and information provided by information systems. All these elements can be modelled in different languages, which are described by certain syntax, semantics and graphical notation. This survey will discuss how to align and map syntax and semantics of the approaches BPDM, BPEL and OWL-S which do not intersect in their purpose and function. In particular the survey analyses the approaches in the sense of feasibility, practicability and completeness when it comes to reference model integration.

1. Introduction

Nowadays, information technology (IT) we find in every business. The use of IT is essential in the business world to communicate and to process the today's amount of data. Businesses have to respond to changes in the market as fast as possible. These changes have influence on the business and IT side, where adaptations have to be made. The management of organizations have to give seriously attention to IT investment, because they are huge and expensive. Thus, the management has to evaluate what the increase in business value is and how to use the IT effective.

IT supports the business to be more effective and efficient. Even more, the function of IT has transformed from a supportive role to a business enabler and contributes therefore to organizational performance [10][11]. But information technology will only add value to the business if it is used effective. On this account business and IT have to be aligned. This means that the right activities have to be supported by IT at the right time. This includes sequences of activities, so called business processes as well.

The difficulty is that there are numerous standards on the business and the IT side, but which mainly support just the goals of one of them. To ensure an alignment of the two sides the standards have to be linked somehow. Or even better would be if a business standard can be transformed to an IT standard or backwards. But obviously, standards have different objectives, concepts, structure, semantic and notations.

Therefore, the source and the destination standards have to be examined for equivalent elements, which could be transformed. If there is no such an equivalent, either a construct has to be worked out, which bypasses the problem and has the same result or the transformation of this element is not possible. As soon as we don't find an equivalent element, the destination model of a standard is compromised, because it can't be ensured that the result of the destination model will be the same like in the source model. Hence, we take basic criteria into

account to exhibit if the transformations are usable. To determine if the destination model after transformation is compromised and still has the same purpose as it had at the source model the criteria completeness and correctness are suitable. Furthermore we analyse the mapping if it is

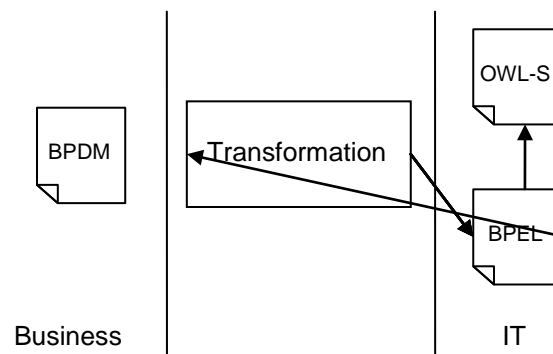


Figure 0: Mappings

applicable and if the process can be automated without losing information. If the last two criteria can't be fulfilled the transformation isn't efficient, because maybe too many adaptations have to be done manually, so that it would be more solid to build the destination model from scratch.

In this paper we examine the transformation process from BPDM to BPEL and BPEL to OWL-S, which is depicted in figure 0. BPDM is the metamodel for one of the leading business process notation BPMN, which is already supported by more than 50 tools [7]. On the IT side we have taken BPEL for the execution of the processes, because it is the most promising one [8] [12]. Unfortunately, BPEL has the drawbacks that it is lacking on semantic enriched information. Thus, the intention should be on the one hand to map the business processes automatically to an executable, semantic enriched language and to transform BPEL to this enriched language. One approach of such a concept is OWL-S, which is based on the leading ontology approach OWL. The mappings are analysed according to the previously mentioned criteria.

The paper is organized as follows: In section 2 we give a brief overview of the examined concepts, namely BPDM, BPEL and OWL-S. Next, section 3 and 4 depict the mapping of the base elements of BPDM and BPEL, respectively. Additionally, limitations of the transformation process and examples of implementations are presented. Finally, section 5 concludes our work and outlines future work.

2. Preliminaries

2.1. BPDM

Interoperability is and always was a main challenge when it comes to information exchange, also for business processes. In a world with almost endless competitors and proprietary software, you may not achieve interoperability without backing out for a moment and define a true normative foundation, which are independent of any form of notation. Normative means standardized. The Business Process Definition MetaModel [17] creates this opportunity for businesses by dictating the XML Metadata Interchange (XMI) format. XMI preserves the intended enactment and execution semantics of the process. In addition, BPDM defines two fundamentally but complementary views of processes – Orchestration and Choreography. Choreography is nothing different than coordination of processes. A process is considered an orchestration of activities. BPDM, in particular the choreography concept, helps to model a Service Oriented Architecture.

Specifically speaking, 4 packages comprise BPDM. Common Abstraction like the Composition and Course model, enriched with the Common Behaviour model and, of course, an activity and an interaction model. Taking these predefined elements into account; one can create any notation type favourable and still compliant with BPDM. But even more important, the modelled processes can still be interchanged and interpreted by other processors, presupposing it supports BPDM and XMI. This means also agility in the process of selecting a product or service sources.

2.2. BPEL

To leverage the power of process modelling from the business, IT-responsible persons require an executable syntax, which a computer system can interpret non-ambiguously. The Business Process Execution Language [1] supports this objective by a block structured notation, well-formatted in XML. Introducing BPEL is one of the initial technical steps in creating a service oriented architecture with coded orchestration and choreography. It goes without saying that this syntax is system independent.

2.3. OWL-S

In business more and more Web Services are employed and are connected to the business processes where they are used. WSBPEL 2.0 [1], Web Services Business Process Execution Language, is the most popular standard in the industry, which can provide the orchestration of Web Services. Unfortunately, this modelling language is lacking of assistance for semantic information. Therefore, OWL-S, Web Ontology Language for Services, was developed to access web resources by content rather than just by keywords [2]. Broadly spoken, OWL-S consists of three types of knowledge: (1) the Service Profile defines what the service provides, (2) the Ser-

vice Model states how the service is used and (3) the Service Grounding declares how Web Services and clients interact with each other.

2.4. WSDL

The Web Service Description Language V2.0 [15] is an XML-based language to describe the interaction with web services. The receiving and sending messages, what the service provides, how to access the service and where to access the service can be described.[15] WSDL doesn't have any semantic enriched information and expresses just a simple behaviour. To provide complex behaviours a language like BPEL has to be used. To provide semantic enriched information and complex behaviour OWL-S can be employed.

3. Mapping BPDM to BPEL

Business Process Definition Metamodel is building on MetaObject Facility [18], and in that way it depicts objects and relationships it finds on the Infrastructure Library [19]. The IL is an abstract package comprised of a subset of the UML class diagram. It represents the core modeling elements that BPDM uses to model their own meta elements.

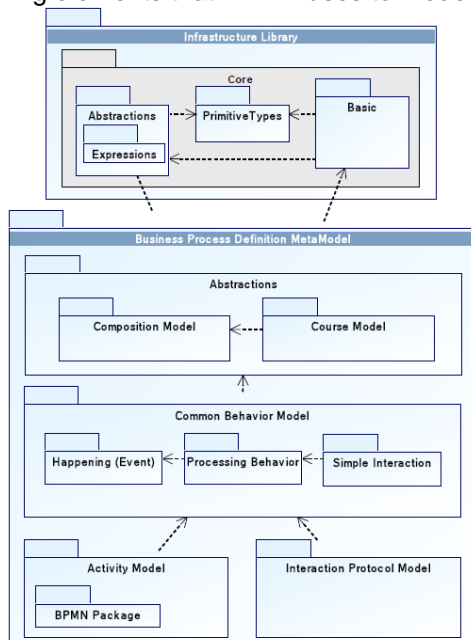


Figure 1: MetaObject Model (based on [17][18][19])

The IL as foundation, BPDM stacks its own abstracts on top of it. Notably, it is the Composition and Course Model. These two basically allow creating individuals and relationships on the MOF M0 layer as recalled [18]. Common behaviour models that are used in processes may be attached to the individuals or vice versa. Strangely enough, the OMG Group did place partial BPMN definition [18] inside the BPDM definition. There is no solid explanation available for this, especially since the BPDM claims to be independent and the foundation for interchangeable and interoperable process notations.

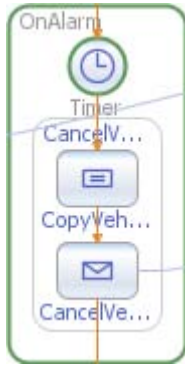
However, since BPDM and BPEL evolved from different backgrounds, conceptual mismatch occur, when trying to align similar elements of the two concepts. Best if they would be equivalent. This is difficult to find, because the paradigm of their structure is different. Textual and graph oriented versus a block structure [17] do not ease the endeavours to establish a fully automated transformation from BPDM to BPEL. The pursuit of automation requires the correctness of the transformation. The logic of the process

must not be compromised at any circumstance or, more specific, with any given complexity. This requirement entails completeness of the BPEL elements you may use to depict a process or a process orchestration. Missing constructs may result in a transformation not applicable.

3.1. OMG's recommended Mapping

Given these criterias, OMG defines the BDPM-BPEL Mapping [19] and thus relieves engineers from mapping themselves.

The *Processor Role* finds its equivalent in the tag <process>. Whereupon the name of the Processor Role goes to the name attributes of <process>. Considering this very simple alignment of a well-structured element, others may be more difficult to map. A time based condition is mapped by the tags <onAlarm>, <for> and <sequence> as seen in Figure 2 and 3. For the ease of imagination, we have chosen the BPMN notation to illustrate.



```

<onAlarm>
  <for>'P0Y0M0DT0H0M20S'</for>
  <sequence name="AirlineCancelSequence">
    <assign name="CopyAirlineCancellation">
      <copy>
        <from>$ReserveAirlineIn.itinerary/ota:ItineraryRef</from>
        <to variable="CancelAirlineIn" part="itinerary"/>
      </copy>
      <copy>
        <from variable="CancelAirlineOut"/>
        <to variable="CancelAirlineOut"/>
      </copy>
    </assign>
    <invoke
      name="CancelAirline"
      partnerLink="Airline"
      operation="cancelAirline"
      portType="ares:AirlineReservationPortType"
      inputVariable="CancelAirlineIn"
      outputVariable="CancelAirlineOut"/>
  </sequence>
</onAlarm>

```

Figure 2: BPMN with Time Condition

Figure 3: BPEL with Time Condition

3.2. The conceptual mismatch

Although the burden of mapping is removed by OMG, some issues in the transformation process from BDPM to BPEL remain. It's not vice versa because BDPM is immensely more verbose - and equipped with alternatives - than PBEL. An alleged impotence to transform denotes impotence in automating the process by a business information engine (e.g. Visual Service Design Tool).

3.2.1. Control Flow mismatch

To compare possible control flows in a process, this survey relies on the expertise of the Workflow Management Coalition (WfMC) [23]. The WfMC has identified all possible workflows and summarised them on their site [23]. According to that listing, the survey makes obvious what can be achieved and what is out of line. In cases of a minus (-), the language is not capable to express the workflow. The plus (+) indicates full capability and plus/minus (+/-) a lack of support. Full capability means it's unambiguously, which is a requirement for automating the transformation process. At the end of the day, a transformation/translation engine, in an integrated development environment (IDE) or in an application server, should only support the patterns where BPDM and BPEL display a plus. Otherwise wrong implications or deadlocks might occur, and the process/workflow is compromised.

Workflow Pattern	BPDM	BPMN	BPEL	Wrkflw Pattern	BPDM	BPMN	BPEL
<i>Basic Control Flow</i>				Implicit Termination	+	+	+
Sequence	+	+	+	<i>Multiple Instances Patterns</i>			
Parallel Split	+	+	+	MI without Synchronization	+	+	+
Synchronization	+	+	+	MI with a priori Design Time Knowledge	+	+	+
Exclusive Choice	+	+	+	MI with a priori Runtime Knowledge	+	+	-
Simple Merge	+	+	+	MI without a priori Runtime Knowledge	+	-	-
<i>Adv. Synchronization</i>				<i>State-Based Patterns</i>			
Multiple Choice	+	+	+	Deferred Choice	+	+	+
Synchronizing Merge	+	+/-	+	Interleaved Parallel Routing	+	+/-	+/-
Multiple Merge	+	+	-	Milestone	+	-	-
Discriminator	+	+	-	<i>Cancellation Patterns</i>			
<i>Structural Patterns</i>				Cancel Activity	+	+	+
Arbitrary Cycles	+	+	-	Cancel Case	+	+	+

Table 1: matches and mismatches of alignment primarily based on [23] and extended with BDMN [17][19]

Table 1 shows BPDM is massive and plain solid when it comes to process modelling. BPMN as a subset is less expressive and BPEL as a block structure language limited to many of the sophisticated workflow patterns.

3.2.2. Process Representation mismatch

There are no fundamental difficulties in mapping a BPEL process to an isomorphic BPMN diagram. In other words, and again, any BPEL process can be visualized as a BPMN diagram without rearranging the flows. But it is not always possible to map a BPMN diagram directly to an isomorphic BPEL process [20]. BPEL does not know the concept of a GOTO as used in some programming languages because it is block oriented and requires a join to synchronize all of the flows. Therefore, arbitrary sequence flows may not be drawn. The following diagram is an example. Since BPEL does not have GOTO constructs, we need to redraw a semantically equivalent process flow.

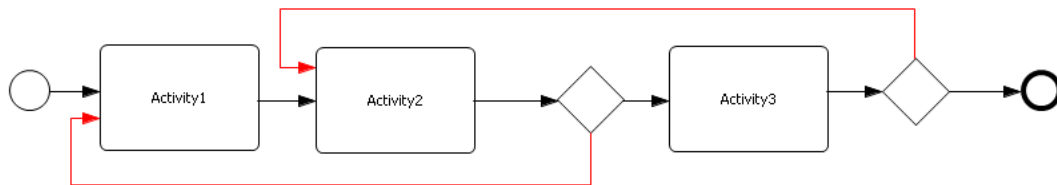


Figure 4: A flow construct in BPMN to rewrite

The two overlapped loops can be redrawn as the following with only one loop before isomorphic mapping to a BPEL process. For what reason do we need to redraw? Again, it's because BPEL block oriented language. Isomorphic in these context means semantically and logically the same. Simply speaking, wrap a condition around every activity to avoid a "GOTO" in order to allow additional process synchronization. This approach is named the Structure-Maximization strategy [21] and basically deals with well-structured pattern-based translation of the BPEL constructs, <sequence>, <flow>, <switch>, <pick> and <while> [21].

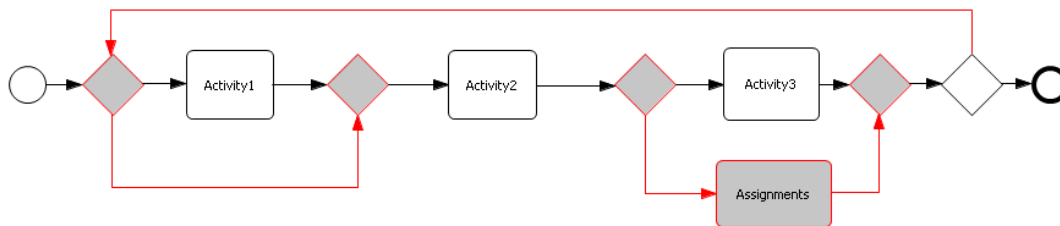


Figure 5: the rewritten flow construct of figure 2 in BPEL

3.3. Case Study

Currently there is no solid software solution that could transform from BPMN to BPEL on the fly. This is due to their different evolution and original objectives. However, many tools, like eclipse [24], are capable of storing BPEL and BPMN files in an XMI in order to interchange; some of them even translate the incoming BPMN to a BPEL file. These are for example NetBeans 6.5 [26], Visual Architect [25] and Eclipse [24]. Unfortunately the only support the very basic workflow patterns in variance. Meaning, for complex workflow one might to rewrite the BPMN graph in order to reach a valid BPEL file. Since these tools know reverse engineering, they may also draw a BPMN graph from a BPEL file at hand, which is as previously mentioned the less challenging direction.

4. Mapping BPEL to OWL-S

Before explaining the equivalent elements in BPEL and OWL-S, we first have a look in what this modelling concepts differ. Obviously, one difference is that OWL-S can deal with semantic enriched information. OWL-S can refer to concepts and individuals of existing domain ontologies. This allows finding, invoicing and composing services automatically.[2] In BPEL the services have to be manually assigned to a process, which is modelled by a partner link. In view of the fact that OWL-S includes semantic information and BPEL doesn't, it becomes clear that the mapping of these two concepts can't be made fully automatic.

The two concepts also differ in their bodywork. In [8] the distinction is that BPEL is workflow language and uses flow to describe the business process while OWL-S converts the process to a hierarchical structure. Therefore, significant changes considering the structure have to be done in the mapping process.

In the next subsections we examine the transformation process according to the criteria introduced in the introduction. Mainly we want to derive the Service Model, Service Profile and Service Grounding from the corresponding BPEL and WSDL file.

4.1. Map Service Model

In the Service Model of OWL-S processes can be defined, which describe how to interact with a service. The process structure and the constructs for designing a process are very similar to BPEL.[5] First we introduce the elements to transform a process. Afterwards we map the (2) basic and (3) structured activities to OWL-S. Basic activities are used to define the elemental steps of a process behaviour, where we can include the data handling.[1] Structured activities can contain basic activities and encode the control-flow logic.[1]

In BPEL the interaction with services is encapsulated in a partner link (figure 6). Thus, for each partner link there is a corresponding operation in WSDL (figure 7). A single interaction to a service can be mapped with assistance of the WSDL file to an atomic process in OWL-S (figure 8). In the example the service will place an order and will respond if the transaction was successful or failed.

```
<pInk:partnerLinkType name="odering">
  <pInk:role name="orderingService"
    portType="InS:orderingType" />
</pInk:partnerLinkType>
```

Figure 6: Partner Link in BPEL (based on [4])

```
<wsdl:portType name="OrderPortType">
  <wsdl:operation name="setOrderingProcess">
    <wsdl:input message="tns:OrderingRequest" />
    <wsdl:output message="tns:OrderingResponse" />
  </wsdl:operation>
</wsdl:portType>
```

Figure 7: WSDL Operation (based on [4])

```
<process:AtomicProcess rdf:about="#setOrderingProcess">
  <rdfs:label>getOrderingProcess</rdfs:label>
  <process:hasInput rdf:resource="#OrderingRequest"/>
  <process:hasOutput rdf:resource="#OrderingResponse"/>
</process:AtomicProcess>
```

Figure 8: atomic process in OWL-S

After seeing how the central element, the process behaviour, can be mapped to OWL-S, we need to transform the basic activities of BPEL to define and form the process steps. To execute the service in OWL-S the "perform" control construct can be used (figure 9). This construct allows sending and receiving messages to and from web services, respectively. The equivalent elements in BPEL are the basic activities receive, reply and invoke. After mapping these three basic activities the URI has to be manually adapted, which specifies where the needed OWL file can be found. This adoption could be supported in several ways by the transformation application.

```
<process:Perform rdf:about="http://example.com/dummyURI.owl#getOrderingProcess"/>
```

Figure 9: atomic process in OWL-S (based on [4])

BPEL allows handling errors. Basic activities like *throw*, *wait*, *exit*, *rethrow* etc. can be implemented directly in the files. In OWL-S there are no equivalent elements to map the basic activities for the error handling. Nevertheless, a simple error handling can be manually applied in OWL-S API, which is a Java API to read, execute and write OWL-S service descriptions.[9]

Additional limitations in the mapping process appear in the data handling. In BPEL variables and data assignments can be done. Again, there is no equivalent control construct to transform the *assign* element directly. Nevertheless, the approach of [4] uses the OWL-S *InputBinding* and *hasDataFrom* constructs, but which can just handle simple data. In figure 10 the service *setOrderingProcess* will be performed with the input of the output of the *SelectionService*. The

SelectionService reads out the selection the customer has made. Again, the URI's have to be adapted to the saving location of the OWL file.

```
<process:Perform rdf:about="http://example.com/URIdummy.owl#setOrderingProcess">
  <process:process rdf:resource="http://example.com/URIdummy.owl#setOrderingProcess"/>
  <process:hasDataFrom> <process:InputBinding>
    <process:toParam rdf:resource="http://service.com/DummyURI.owl#OrderingService.wsdl%23
      inputString"/>
  <process:valueSource> <process:ValueOf>
    <process:theVar rdf:resource="http://service.com/DummyURI.owl#SelectionService.wsdl%23
      return"/>
  <process:fromProcess rdf:resource="http://example.com/URIdummy.owl#setSelectionProcess"/>
</process:ValueOf> </process:valueSource>
</process:InputBinding> </process:hasDataFrom>
</process:Perform>
```

Figure 10: simple data handling in OWL-S

After mapping a process behaviour and the basic activities, we have to structure the steps and the sequence of the processes. Hence, we have to transform the structured activities of BPEL like *sequence*, *flow*, *If-Then-Else*, *while*, *repeatUntil* etc. to OWL-S.

For the mapping of the BPEL element *sequence* exists in OWL-S an equivalent control construct of the same name. The element will perform the on processes in sequence, one after another.

The elements *while* and *repeatUntil* can be mapped to equivalent OWL-S element *repeat-while* and *repeat-until* control construct, respectively. The elements iterate if a condition is true or false. The difference is that *repeat-while* tests the condition at the beginning, whereas *repeat-until* does it at the end. Therefore, the repeat-until always performs once.

In BPEL v2.0 the *switch* activity was replaced by the *If-Then-Else* construct and can therefore be mapped directly to OWL-S element of the same name. If a BPEL file of an earlier version has to be mapped to OWL-S, the *If-Then-Else* element can be used as well [4].

To define the concurrent start of activities in a process, BPEL uses the *flow* element. In OWL-S the equivalent element to provide concurrency is the control construct *split*. The mapping of the element is shown in the example (figure 11 and 12).

In the example an order is received. Afterwards at the same time the process to send a confirmation to the customer (*setOrderConfirmation*) and the assembling of the order (*setAssemblingOrder*) is started. To provide synchronisation, OWL-S provides the *split-join* control construct. Using *split* and *split-join* partial synchronisation can be modelled. The mapping of synchronisation from BPEL to OWL-S need continuing research and is therefore not further explained.

```
<invoke ... operation="getOrder" ... />
<flow>
  <invoke ... operation="setOrderConfirmation" ... />
  <invoke ... operation="setAssemblingOrder" ... />
</flow>
```

Figure 11: BPEL flow activity statement

```
<process:Perform>
  <process:process rdf:resource="http://example.org/DummyURI.owl#setOrderProcess"/>
</process:Perform>
<process:Split-Join> </process:components>
  <process:Perform>
    <process:process rdf:resource="http://example.org/DummyURI.owl#setOrderConfirmationProcess"/>
  </process:Perform>
  <process:Perform>
    <process:process rdf:resource="http://example.org/DummyURI.owl#setAssemblingOrderProcess"/>
  </process:Perform>
</process:components></process:Split-Join>
```

Figure 12: OWL-S split control construct statement

As we saw, there are some limitations in the transformation process regarding the completeness. For the most of the elements in BPEL, equivalent control constructs in OWL-S could be found. There are limitations in mapping the error handling, data handling, and synchronisation and to make condition statements [8]. Still, simple BPEL elements can be transformed correct.

4.2. Map Service Profile

The Service Profile can be divided into the three aspects classification, non-functional and functional [13]. To classify a service, references to a taxonomy can be made. In the non-functional aspect we can define additional information, like service name, textual description and contact information. Inputs, outputs, preconditions and results (IPORs) are defined in the functional aspect. The first two aspects aren't covered by BPEL and can't be mapped to OWL-S, but can be adapted manually after the transformation process. Elements of the third aspect can be mapped, but because BPEL is a workflow language, it is nowhere clearly stated what are the IPOR's. A solution to this problem is to find the first BPEL activity which receive a message from the outer world and the last activity which sends a message to the outer world. These activities are then mapped to the control construct *hasInput* and *hasOutput* of OWL-S, respectively. Because BPEL doesn't define preconditions and results, the control construct in OWL-S are just omitted, what results in no loss of information. After the mapping the URI's have to be adapted, as we have seen in previous matters.

4.3. Map Service Grounding

The grounding specifies the access to the corresponding WSDL file and defines how to access the service. [2] The grounding includes several control constructs to define where the WSDL file is located, how to bind the service and which operations it has. This information can be transformed from the equivalent elements in BPEL. The URI's have to be adapted manually. Additionally, OWL-S supports to define concrete message formats by using XSL [14]. This XSL constructs have to be done manually.

4.4. Case Study

In this section we have a look at the BPEL4WS2OWL-S Mapping Tool V1.1 [3] and examine if the mapping from BPEL to OWL-S can be applied in business.

The tool supports the transformation of simple BPEL files with the aid of the corresponding WSDL to the OWL-S service profile, service model and service grounding. The application is very simple to handle. The user has to import the BPEL and WSDL files and by pressing a button the OWL-S files are created. But the tool has some limitation, which makes it impossible to transform complex BPEL files. As we have seen in the previous sections data handling, error handling and synchronisation of processes aren't supported. Additionally, after the transformation there have to be done manual adaptations like changing the URI's or specifying the detailed service profile. Therefore, the tool isn't applicable in business. Tools like OWL-S Editor [6] provide functions to easily create new OWL-S files. Currently, creating new files is nearly as fast as mapping the BPEL files to OWL-S.

5. Conclusion

Our research shows clearly that the transformation from BPDM to BPEL and BPEL to OWL-S do not satisfy the above mentioned criteria of completeness, correctness and the ability to automate and apply for practical purpose. Therefore, this approach does not allow creating an executing language as BPEL - derived from the BPDM. This entails that researcher must look into other solutions in order to retain the semantics of a process modelling language. Further work in these fields, we suggest, might look into the transformation of BPDM to OWL-S, or to the Web Service Modeling Ontology (WSMO), which is

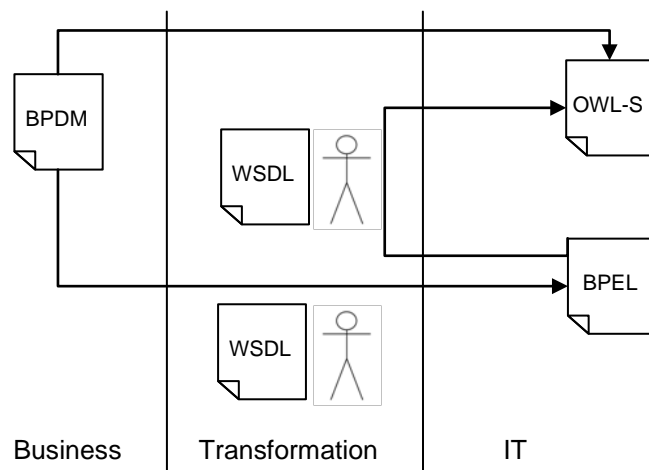


Figure 13: Mappings

shown in figure 13. And once again, the transformation is expected to be correct, complete, capable of automating the transformation process and applicable. If the translation is possible to these terms, an organization will benefit of a fully automated process. A process that doesn't require change if for example webservice addresses change. A process that has no human interaction as long the business logic remains unchanged. And best of it, a process that requires no IT expert to modify the workflow code. The business person himself may change the process whenever the situation demands. Business wins agility that creates a true added-value, especially in rapidly changing industries. It goes without saying that this is certainly an area business can expect a return on investment.

References

1. OASIS, 2007. *Web Services Business Process Execution Language Version 2.0* [Online]. Available from: <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf> [Accessed 3 December 2008].
2. The World Wide Web Consortium, 2004. *OWL Web Ontology Language Semantics and Abstract Syntax* [Online]. Available from: <http://www.w3.org/Submission/OWL-S/> [Accessed 2 December 2008].
3. Aslam, M.A. *BPEL4WS 2 OWL-S Mapping Tool V1.1* [Online]. Available from: <http://bpel4ws2owls.sourceforge.net/> [Accessed 19 December].
4. Aslam, M. A., Auer, S., Shen, J., and Herrmann M., 2006. *Expressing Business Process Model as OWL-S Ontologies*. Wollongong: University of Wollongong.
5. Feldkamp, D. and Singh, N., 2008. *Making BPEL flexible*. Olten: University of Applied Sciences Northwestern Switzerland.
6. Scicluna, J., Abela, J., 2004. *OWL-S Editor to Semantically Annotate Web-Services* [Online]. Available from: <http://staff.um.edu.mt/cabe2/supervising/undergraduate/owlseeditFYP/OwlSEdit.html> [Accessed 19 December].
7. Object Management Group, 2008. *BPMN Activities* [Online]. Available from: <http://www.bpmn.org/workinggroup.htm> [Accessed 10 December 2008].
8. Shen, J., Yang, Y., Zhu, C., Wan, C., 2005. *From BPEL4WS to OWL-S: Integrating E-Business Process Descriptions*. Melbourne: Swinburne University of Technology.
9. Mindswap, 2006. *OWL-S API* [Online]. Available from: <http://www.mindswap.org/2004/owl-s/api/> [Accessed 15 December 2008].
10. Devaraj, S., and Kohli, R., 2003. *Performance Impacts of Information Technology: Is Actual Usage the Missing Link*. Management Science.
11. Mukhopadhyay, T., Kekre, S., and Kalathur, S., 1995. *Business Value of Information Technology: A Study of Electronic Data Interchange*. MIS Quarterly.
12. Ouyang, C., van der Aalst, W. M. P., Dumas, M., ter Hofstede, A. H. M., 2006. *Translating BPMN to BPEL*. Brisbane: Queensland University of Technology.
13. Martin, D., Burstein, M., McDermott, D., McIlraith, S., Paolucci, M., Sycara, K., McGuinness, D, Sirin, E. and Srinivasan, N., 2007. *Bringing semantics to web services: The OWL-S approach*.
14. Martin, D., Burstein, M., Lassila, O., Paolucci, M., Payne, T., McIlraith, S. *Describing Web Services using OWL-S and WSDL* [Online]. Available from: <http://www.daml.org/services/owl-s/1.1/owl-s-wsdl.html> [Accessed 18 December 2008].
15. The World Wide Web Consortium, 2007. *Web Services Description Language (WSDL) Version 2.0 Part 0: Primer* [Online]. Available from: <http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626/> [Accessed 18 December 2008].
16. OMG BPDM Group. *The Benefits and Objectives of BPDM* [Online]. Available from: http://www.waria.com/Documents/BPDM_Executive_Overview.pdf [Accessed 12 December 2008].
17. OMG, 2008. *Business Process Definition MetaModel, Volume I: Common Infrastructure* [Online]. Available from: <http://www.omg.org/docs/formal/08-11-03.pdf> [Accessed 24 December 2008].
18. OMG, *MetaObject Facility* [Online]. Available from: <http://www.omg.org/mof/> [Accessed 30 December 2008].

- 19 OMG, 2008. *Business Process Definition Model, Volume II: Process Definitions* [Online]. Available from: <http://www.omg.org/docs/formal/08-11-04.pdf> [Accessed 28 December 2008].
- 20 Yi Gao, 2006. *BPMN-BPEL Transformation and Round Trip Engineering* [Online]. Available from: http://www.eclarus.com/resources/BPMN_BPEL_Mapping.pdf [Accessed 21 December 2008].
- 21 Schumm, D., 2008. *Graphische Modellierung von BPEL Prozessen unter Verwendung der BPMN Notation* [Online]. Available from: ftp://ftp.informatik.uni-stuttgart.de/pub/library/medoc.ustuttgart_fi/DIP-2720/DIP-2720.pdf [Accessed 2 Jan 2008].
- 22 Recker, J., Mendeling, J., 2007. *On the Translation between BPMN and BPEL* [Online]. Available from: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.6893>, [Accessed 1 Jan 2008].
- 23 Workflow Management Coalition. *Terminology and glossary* [Online]. Technical Report Document Number WfMC-TC-1011, Issue 3.0, 1999. Available from: http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf [Accessed 1 Jan 2008].
- 24 Eclipse organisation, 2006. *GMF project Requirements* [Online]. Available from: <http://www.eclipse.org/modeling/gmf/requirements.php#tools> [Accessed 1 Jan 2008].
- 25 Visual Architect, 2008. *Visual Paradigm* [Online]. Available from: <http://www.visual-paradigm.com/product/bpva/features/> [Accessed 1 Jan 2008].
- 26 NetBeans, 2008. *Enterprise Service Oriented Architecture Project Home* [Online]. Available from: <http://enterprise.netbeans.org/soa/> [Accessed 1 Jan 2008].
- 27 Ouyang, C., Dumas, M., ter Hofstede, A. H. M., van der Aalst, W. M. P., 2007. *Pattern-based Translation of BPMN Process Models to BPEL Web Services International Journal of Web Services Research* [Online] Available from: <http://is.tm.tue.nl/staff/wvdaalst/publications/z9.pdf> [Accessed 1 Jan 2008].