Bridging the Gap between BPMN and WS-BPEL. M2M Transformations in Practice[1]

Pau Giner, Victoria Torres, Vicente Pelechano

Department of Information Systems and Computation
Technical University of Valencia
46022 Valencia, Spain
{pginer, vtorres, pele}@dsic.upv.es

Abstract. The Web is being consolidating as the main platform for the development of applications. Moreover, these applications are not conceived just as isolated systems. This fact implies that the requirements that Web applications should satisfy are now more challenging than before. One important requirement for these systems is to provide support for the execution of business goals expressed by means of Business Process definitions. From the Web Engineering area, several methods have provided a solution to cope with this requirement. In this work we present, within the context of the OOWS Web Engineering method, how business process definitions are transformed into executable process definitions by the application of model-to-model transformations. To accomplish this goal, this work has been developed in the context of the Eclipse environment jointly with the BABEL project.

1 Introduction

Web applications are no longer conceived just as systems to perform CRUD functions over the persistence layer. In fact, the possibilities that bring the environment in which these applications live widen the kind of systems being build as well as introduce new challenges such as security, reliability, integration, etc.

One of the main advantages introduced by the Internet is that "services" are available 24x7. This fact allows service providers to reach a larger community of customers. Moreover, when these services are offered using a standard technology the potential number of customers can grow easily. In this direction, Web services were built as the standard technology to provide functionality over the Web.

However, the great potential of services does not limit to the use of services as units. In contrast, it is the service composition what brings value to them. Service composition usually involves the interaction between different partners, some of them behaving as clients and others as providers. Then, if we go one step forward, we can see service compositions as business processes, where different services provided by different partners are put together to accomplish certain agreed goals.

[1] This work has been developed with the support of MEC under the project DESTINO TIN2004-03534 and cofinanced by FEDER.

In a previous work [11] we presented an extension to the OOWS [9] Web Engineering method to provide support for the generation of Business Process Driven Web applications. This extension embraced mainly the Navigational model. The main goal of this work was to obtain from a business process definition the Navigational model necessary to provide support to the original processes. Moreover, as these processes can extend in time, we decided to introduce into the architecture of the generated applications a process engine that was in charge of driving processes during their life. Therefore, we need to transform these business process definitions into a format that could be executed by the engine.

Moreover, following a Model Driven Approach for the construction of these applications allows us to define them in a technological independent way (in terms of the service composition) as well as to perform separation of concerns. In this case, as we bet on Web services the independence is relegated to service composition. From service compositions defined in the BPMN [5] notation we could then transform it into different process executable languages. In this work we focus on the generation of WS-BPEL [6].

The main contribution of this work is to present the application of the MDA approach within a Web Engineering method for the construction of Business Process driven Web Applications. In particular, this work focuses on the task of translating business processes defined graphically in the BPMN notation (defined at the PIM) into a specific language such as WS-BPEL (placed at the PSM level). Moreover, this work has been developed within the Eclipse and BABEL [2] projects.

The remainder of the paper is structured as follows. Section 2 provides a revision over the related work developed within in the Web Engineering area. Section 3 puts into context the work developed and presents the tools used to accomplish it. Section 4 provides a brief overview over the BPMN language (the language used in this work for service composition). Section 5 presents step by step the process followed to extend an existing to tool to provide full transformation from BPMN to WS-BPEL. Section 6 presents some conclusions about the experience of this work. Finally, two appendixes are included to show both, the schema generated for the extended tool and the ATL [1] transformations implemented for this purpose.

2 Related Work

Web Engineering methods provide modeling mechanisms (supported in some cases by tools) to overcome the development process of web solutions. Due to the inherent dynamism of the Web, most of these methods have evolved to provide support to the new arising requirements. As a result of these requirements a broader range of systems are considered by these methods. Within this range we find process-driven web applications.

In this context, some of the existing proposals developed within the Web Engineering area have coped with the issue of integrating business process with navigation. The solutions provided by these methods can be divided into two groups. In the one hand, some of them propose introducing business process primitives into the naviga-

tional model. Within this group we find proposals such as OOHDM [10], UWE [8] or WSDM [7]. On the other hand, others propose simply modeling the navigation that occurs during the process execution as if it were pure navigation. In this group we find the OO-H [8] proposal. All these proposals make use of a process notation such as UML Activity Diagrams or Concur Task Tree to define process requirements. However, it is important to note that the navigation allowed within a business process execution is completely driven by the process and not by the user. This means that it is not necessary to include in the Navigational model the process flow. In fact its inclusion can complicated considerately the understanding and modeling of the navigational model. Nevertheless, this do not implies that no navigation has to be defined for business process execution. In fact, it can be desirable to improve or complete the content of the navigational nodes (web pages) in order to generate better user interfaces.

Regarding workflows or long-running processes, WebML [4] and Hera [3] provide modeling mechanisms to support this kind of processes. The modeling of this kind of processes involves that multiple process and activity instances can be handled by users. Moreover, different users behaving with different roles are in charge of performing certain process tasks. In the one hand, the webML approach introduces a process reference model into its conceptual model. This process reference model is interconnected with the application data model and the user model and it is used to control the state of cases and processes. This proposal also introduces primitives to model the navigation that occurs during process execution. However, these are not separated from the ones that refer to pure navigation. As a result, navigational models can get complicated not only when the size of processes is considerable or the process grows but also if the control flow includes too many forks.

Similarly, Barna et al. in [3] propose a specification method for the automatic generation of web models from a workflow model. This proposal takes into account workflow processes providing a solution at the modeling level by introducing a task and workflow modeling phases. This proposal also considers the asynchrony that appears in workflow processes. To overcome it they present a mechanism based on message queues to handle multiple workflow instances. Again, the navigation of the workflow is moved to the navigation structure what complicates the definition of the navigation.

Our proposal tries to introduce modeling mechanisms for developing both short and long-running processes. Moreover, we have tried to minimize the impact that this new mechanisms can have over the remainder models. For this reason, we introduced the Business Process Model (BPM) that allows us defining the set of business processes that govern the organization. The Navigational Model is only used to specify the navigational contents that are going to be included in the implemented GUIs. No navigation is again defined in this model. The navigation that occurs during process execution is fixed and the user only has to follow it.

3 Work Context

This work has been developed as a part of a bigger project aimed at the development of Business Process driven Web applications. This bigger project involves the development of Web applications based on the MDA approach. Next two subsections are dedicated (1) to present an overview of the whole project pointing out the part covered in this work and (2) to reference the tools that have been used in its development.

3.1 Project Overview

As we have mentioned previously, the project has been conceived taking into account the MDA approach. Following this philosophy, the whole system is specified in a technological independent fashion by means of different models which represent the different aspects that characterize this kind of Web applications. Fig. 1 shows both the models included in the proposal (including the dependences between them) and the transformations required to evolve this specification into new models or even final code (depending on the case).

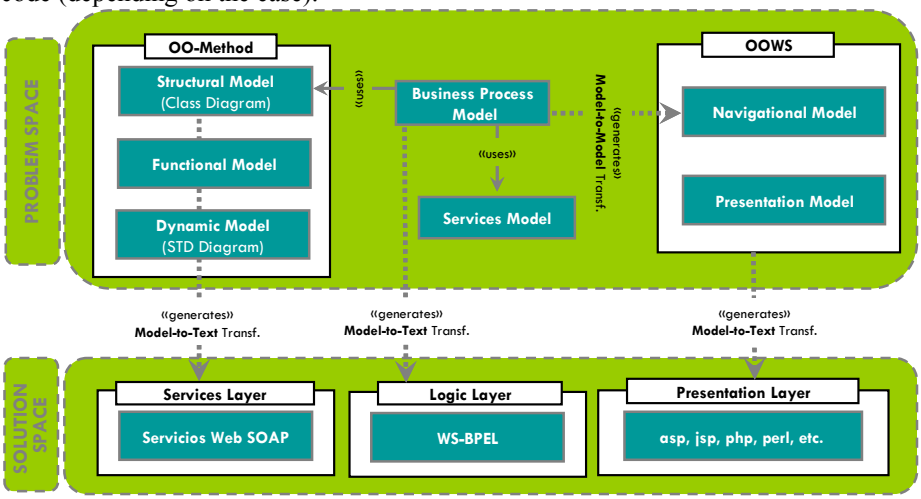

Fig. 1 Project Overview

This figure shows that the Business Process Model (BPM) is defined using functionality that is defined both in the Structural Model and in the Services Model. This allows the composition at the modeling level of internal functionality and functionality that is "imported" from external partners. The following paragraphs provide a rough explanation of these models.

The OO-Method (Object Oriented Method for Software Development) models specify the structural and functional requirements of dynamic applications. These models are:

- *the Structural Model*, which defines the system structure (its classes, operations and attributes) and relationships between classes by means of a Class Diagram.
- *the Dynamic Model*, which describes (1) the different valid object-life sequence for each class of the system using State Transitions Diagrams and (2) the communication between objects by means of Sequence Diagrams.
- *the Functional Model*, which captures the semantics of the state changes to define service effects using a textual formal specification.

The Services and Business Process models were introduced to specify the interaction with external partners.

- *the Services Model*, which brings up external services (functionality provided by external partners) into the modelling level in order to manage them more easily [12].
- *the Business Process Model*, which specify by means of BPMN diagrams a set of processes in which can intervene both functionality provided by the local system as well as functionality provided by external partners. The activities that made up these processes represent functionality defined both in the Structural Model and in the Services Model.

The OOWS (Object Oriented Web Solutions) models were introduced in order to extend conceptual modeling to Web environments. These models are:

- *the User Model*, which defines the kind of users that are going to interact with the web application. Moreover, it defines inheritance relationships between them.
- *the Navigational Model* allows us defining appropriate system views (in terms of data and functionality) for each kind of user defined in the User Model. This model was extended in a previous work [11] with a set of new primitives in order to integrate business process execution within the Navigation.
- *the Presentation Model* allows us to model presentation requirements (information paging, layout and ordering criteria) of the elements defined in the Navigational Model.

The whole project relies both on transformations between models (to move knowledge between different aspects) and between model(s) and text (to generate an equivalent/compliance representation in terms of an implementation language). In particular, the part of the project that has been implemented and presented in this work refers to the Model-to-Text transformation that moves process definitions represented in the BPM into the WS-BPEL executable language.

3.2 Technological Context

For the development of this work we have made use of a set of tools most of them included in the Eclipse project and that are commented in the following paragraphs.

The *Eclipse Modeling Framework*[2] (EMF) is the basis of several modeling projects developed by the Eclipse community. EMF includes tools for the generation, edition and serialization of models conforming to Ecore metamodels (an implementation of the OMG's Essential MOF to represent metamodels).

The necessity of operations to work with models comes from the fact that the Model Driven Development approach considers models as first-class citizens. The *Atlas Transformation Language*[3] (ATL) was defined to cope with operations referred to model transformation. It allows the definition of transformation rules for the creation of one or more output models from several input models. Moreover, the scope of model transformation provided by this language is wide being quality improvement, model refinement or model merging examples of some applications. The concept of cartridge, a metamodel representing a technology and the corresponding code generation in conjunction, permits the usage of model transformation to generate final artifacts and helps to maintain the rationale of the generation in a model-to-model transformation. With this approach, in opposition to a direct model-to-code transformation, the artifacts involved in the development are maintained at the modeling level.

XML Schemas are used to define XML-based formats, deriving an Ecore metamodel from them allow the definition of Platform Specific Models (PSM). EMF permits the generation of Ecore metamodels from XML Schemas. Models conforming the generated metamodel are, when serialized, valid according to the schema. *Web Tools Platform*[4] (WTP) project offers several editors for different web-related formats to ease Web applications and Web Services development. The XML tools have been used to define XML Schemas and test them.

The *Babel BPMN2BPEL*[5] *tool* is a java application to transform BPMN diagrams into WS-BPEL definitions. BPMN is a graphical notation and has no defined textual format, so the tool input format is a concrete textual representation with no formal description. The lack of a model behind the input format prevents integration of the tool in a model driven environment. As the format is XML based, the definition of its supported XML Schema and the equivalent Ecore metamodel enables us (1) the creation of models and (2) the automatic code generation for the underlying format. Moreover, it constitutes a cartridge usable at the modeling level.

The *SOA Tools Platform (STP) project* aim is to offer a framework and a set of tools to enable the development of SOA based applications. One of its sub-projects[6] consists on the definition of a graphical editor to create BPMN diagrams. The editor is based on Graphical Modeling Framework (GMF) and the BPMN metamodel has been defined in Ecore, that enables its usage with EMF-based tools. Business Processes are modeled using the BPMN graphical editor included in the STP project and a mapping targeting the platform specific model of Babel tool will be defined.

---

[2] http://www.eclipse.org/modeling/emf/

[3] http://www.eclipse.org/m2m/atl/

[4] http://www.eclipse.org/webtools/main.php

[5] http://www.bpm.fit.qut.edu.au/projects/babel/tools/

[6] http://www.eclipse.org/stp/bpmn

4 Using BPMN for Service Composition

As we have mentioned previously, service composition provide an added value to the companies that provide them. Moreover, the execution of compositions that include services that involve human participation can take a long time for being completed and then it requires keeping the state of the composed service. To accomplish this task we rely on the use of a process engine.

In the following subsections we present the languages used to define and execute service compositions.

4.1 BPMN to define Service Compositions

There are available several notations (such as UML Activity Diagrams, UML EDOC Business Processes, IDEF, ebXML BPSS or BPMN among others) that can be used to model BPs. In particular, we are going to use the BPMN notation because it provides a mapping between the graphics of the notation to the underlying constructs of an execution language, in particular to the WS-BPEL language, what makes this notation a good candidate to be used. This notation is designed to cover a wide range of type of diagrams (such as "high-level private process activities", "detailed private business processes", "detailed private business processes with interactions to one or more external entities" or "two or more detailed private business process interacting" among others). However, as our goal is to obtain those software components that implement these BP definitions, we are going to use the notation for the design of "detailed private business processes with interactions to one or more external entities". It is important to make this clear in order to obtain, after the application of the transformation rules, a running Web Application solution.

4.2 WS-BPEL to Execute Service Compositions

The growth in the adoption of the Web service technology made us to consider services composition languages that were based on this technology. For this purpose, the OASIS consortium is been developing Web services standards to cover for instance issues such as security or e-business. One of these standards is the WS-BPEL, which allows describing the behavior of a business process based on interactions between the process and its partners through Web Service interfaces.

5 Bridging the Gap between BPMN to WS-BPEL

To achieve the goal established in this work (the translation between business processes definitions represented in the BPMN notation into WS-BPEL executable definitions) we have made use of the BPMN2BPEL tool. In fact, we have extended this tool to provide support to the whole transformation.

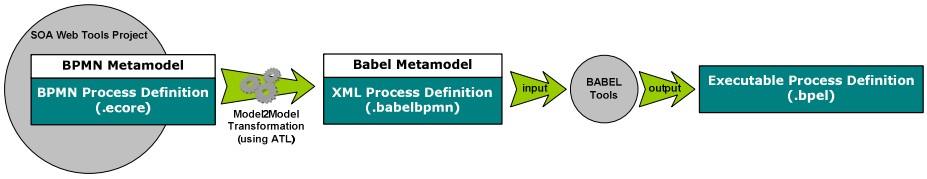

Fig. 2 Process Overview

This tool is part of the process transformation tools developed in the BABEL project (project developed by the Business Process Management group at QUT). However, although this tool translates process models into process definitions represented in WS-BPEL we wanted to provide the transformation directly from the BPMN notation to WS-BPEL. Moreover, as these transformations represent just a part of a bigger project, which is being developed within the Eclipse environment, we wanted to integrate them as well in the same environment. Fig. 2 depicts graphically the process followed to accomplish these goals.

5.1 XML Schema Definition for the BPMN2BPEL tool

BPMN is a graphical notation that lacks of a standard textual representation. Then, the format used by the BPMN2BPEL tool is a particular XML application with no definition of its grammar. In order to make the tool more interoperable it is desirable to have, an XML Schema that represents the model behind the data. Although there are several options to define XML applications (DTD, XML Schema, Relax NG, Schematron, RDF and the like), we decided to use XML Schema because of its tight integration with the EMF tools. XML Schema can be converted to an Ecore metamodel and the instances of this model can be converted back again into an XML conformant with the schema.

First of all we needed to know the schema used by the BPMN2BPEL tool to represent processes. Based on the suit of examples attached to the tool we could extract and produce the process schema. The complete generated schema is included in the Appendix A. This schema defines three elements which are nodes, arcs and code. The first two elements refer to activities and flows respectively. The latter element allows the definition of code that is directly copied into the generated WS-BPEL file.

5.2 Transform the XML schema into the Ecore format

Then, to define transformations between the original BPMN model and the process model used by the BPMN2BPEL tool we have to transform the latter into the Ecore Metamodel (which is used by EMF-based tools and allows us to manipulate it properly).

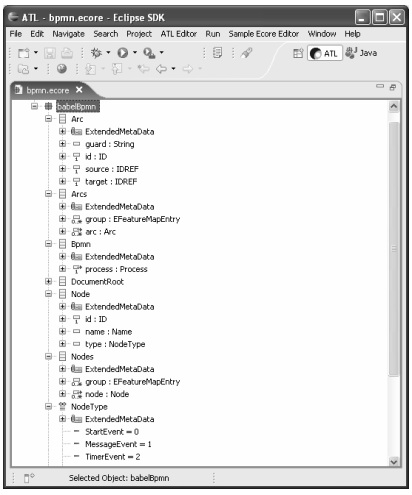

Fig. 3 BPMN2BABEL input model represented in the Ecore Metamodel

Fig. 3 depicts graphically a screenshot with an excerpt of the BPMN2BABEL input model represented now in the Ecore Metamodel.

5.3 Define model-to-model transformations

Once we have both, source (BPMN) and target (process definition used by the BPMN2BPEL tool) metamodels represented in the Ecore Metamodel we could proceed to define the model-to-model transformations. To perform this transformation we have used ATL as the transformation language. Although this language is not fully compliance with the QVT standard it is enough mature and reliable to accomplish the model-to-model transformations required in this project.

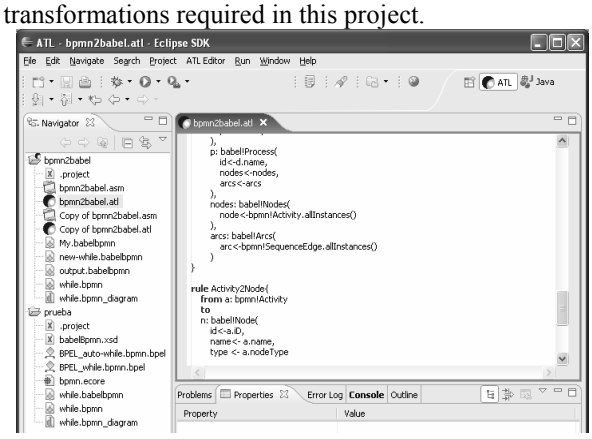

Fig. 4 Screenshot with ATL Transformations

Fig. 4 depicts graphically a screenshot of the implemented model-to-model transformations to obtain the model accepted by the BPMN2BPEL tool. Appendix B includes all the ATL transformations defined in this work.

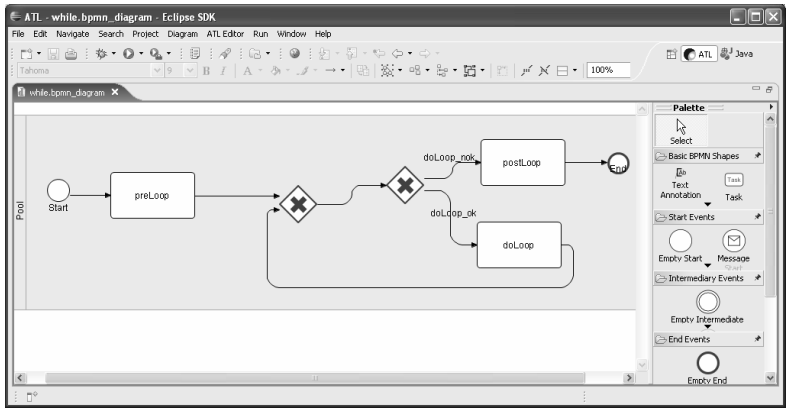

Fig. 5 Business Process Model defined using the BPMN editor provided within the STP project

Finally, the application of these transformations allows us to use the BPMN2BPEL tool to obtain a process definition initially modelled following the BPMN notation (Fig. 5) into an executable definition in WS-BPEL.

6 Conclusions

The application of a Model Driven Approach has proven satisfactory in several aspects. In the one hand reduces the risk of manual errors (in this case the definition of BPMN models was correct since there is a metamodel behind the scenes that enforces a consistent definition them). On the other hand, the use of a graphical tool allows a more comfortable definition of BPMN models.

Moreover, the integration of tools at the modelling level allows the abstraction of technological details and decouples the concepts from its physical representation. Defining the bridge between models with a model-to-model transformation permits the creation of simple mapping rules only dependent of both metamodels. The transformation rules are explicit statements of the equivalence of the metamodels in a technological agnostic way, thus the maintainability of the rules is better than ad-hoc export solutions.

The paper has presented a real application of the MDA approach in the scope of the Web engineering area. In this case, model-to-model transformations have been implemented in the ATL language, which has been proved adequate to achieve the proposed goals. Moreover, the use of some of the projects developed under the Eclipse community such as STP, WTP, EMF and M2M, have make possible its successful development.

Similar to these transformations, the rest of the project involves both model-to-model transformations defined in ATL (to obtain navigational models from business process descriptions) and model-to-text transformations to obtain the executable code that satisfies the system defined at the modeling level.

References

1. ATL: The Atlas Transformation Language. http://www.sciences.univ-nantes.fr/lina/
2. BABEL Project. Expressiveness Comparison and Interchange Facilitation between Business Process Execution Languages. http://www.bpm.fit.qut.edu.au/projects/babel/
3. Barna, P., Frasincar, F., Houben, G.J.: A Workflow-driven Design of Web Information Systems. In: International Conference on Web Engineering, ICWE2006, Palo Alto, USA, 11-14 July 2006, p. 321-328, 2006, ACM.
4. Brambilla, M., Ceri, S., Fraternali, P., & Manolescu, I. (2006). Process Modeling in Web Applications. ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 15, issue 4. October 2006.
5. Business Process Modeling Notation (BPMN) Version 1.0 - May 3, 2004
6. Business Process Execution Language for Web Services version 1.1. Feb 2005. http://www-128.ibm.com/developerworks/library/specification/ws-bpel/
7. De Troyer, O., & Casteleyn, S. (2003). Modeling Complex Processes for Web Applications using WSDM. Paper presented at the Third International Workshop on Web-Oriented Software Technologies, Oviedo, Asturias.
8. Koch, N., Kraus, A., Cachero, C., Meliá, S.: Integration of Business Processes in Web Application Models. Journal of Web Engineering, vol. 3, no.1 pp. 022-049, May 2004.
9. Pastor, O., Fons, J., Abrahao, S., Pelechano, V.: Conceptual Modelling of Web Applications: the OOWS approach. Web Engineering. In: Mendes, E., Mosley, N. (eds), Springer 2006, pp. 277-302
10. Schmid, H. A., Rossi, G.: Modeling and Designing Processes in E-Commerce Applications. IEEE Internet Computing, vol. 8, no. 1, pp. 19-27, January/February, 2004.
11. Torres, V., Pelechano, V.: Building Business Process Driven Web Applications. In: Dustdar, S., Fiadeiro, J.L., Sheth, A. (eds.): Business Process Management. 4th International Conference, BPM 2006. Lecture Notes in Computer Science, Vol. 4102, Springer Berlin / Heidelberg (2006) 322-337
12. Torres, V., Pelechano, V., Ruiz, M., & Valderas, P. (2005). A Model Driven Approach for the Integration of External Functionality in Web Applications. The Travel Agency System. Paper presented at the International Workshop on Model Driven Web Engineering. Sydney, Australia.

Appendix A

This appendix provides a complete listing of the XML Schema used by the BPMN2BABEL tool as input format for business process definition.

```
<schema
   targetNamespace=http://www.bpm.fit.qut.edu.au/projects/babel/bpmn
    elementFormDefault="qualified"
    xmlns=http://www.w3.org/2001/XMLSchema
    xmlns:tns="http://www.bpm.fit.qut.edu.au/projects/babel/bpmn">
```

```
 <element name="bpmn" type="tns:Bpmn"/>
  <complexType name="Bpmn">
   <sequence>
    <element name="process" type="tns:Process"/>
   </sequence>
  </complexType>

  <complexType name="Process">
   <sequence>
    <element name="code" type="anyType" maxOccurs="1" minOccurs="0"/>
    <element name="nodes" type="tns:Nodes"/>
    <element name="arcs" type="tns:Arcs"/>
   </sequence>
   <attribute name="id" type="ID" use="required"/>
  </complexType>

  <complexType name="Nodes">
   <sequence maxOccurs="unbounded" minOccurs="0">
    <element name="node" type="tns:Node"/>
   </sequence>
  </complexType>

  <complexType name="Arcs">
   <sequence maxOccurs="unbounded" minOccurs="0">
    <element name="arc" type="tns:Arc"/>
   </sequence>
  </complexType>

  <complexType name="Arc">
   <attribute name="id" type="ID" use="required"/>
   <attribute name="source" type="IDREF" use="required"/>
   <attribute name="target" type="IDREF" use="required"/>
   <attribute name="guard" type="string" use="optional"/>
  </complexType>

  <complexType name="Node">
    <attribute name="id" type="ID" use="required"/>
    <attribute name="name" type="Name" use="optional"/>
    <attribute name="type" type="tns:nodeType" use="optional"/>
  </complexType>

  <simpleType name="nodeType">
    <restriction base="string">
      <enumeration value="StartEvent"/>
      <enumeration value="MessageEvent"/>
      <enumeration value="TimerEvent"/>
      <enumeration value="XOR-Join"/>
      <enumeration value="EB-XOR-Join"/>
      <enumeration value="AND-Join"/>
      <enumeration value="AND-Split"/>
      <enumeration value="EndEvent"/>
      <enumeration value="XOR-Split"/>
      <enumeration value="EB-XOR-Split"/>
    </restriction>
  </simpleType>
</schema>
```

Appendix B

This appendix provides a complete listing of the ATL Transformations to move from the BPMN to BABEL input format.

```
module bpmn2babel;
create OUT : babel from IN : bpmn;

helper context bpmn!Activity def: isStartEvent:Boolean=
	let types: Sequence(bpmn!ActivityType)=
		Sequence{	#EventStartEmpty,
					#EventStartMessage,
					#EventStartRule,
					#EventStartTimer,
					#EventStartLink,
					#EventStartMultiple}in

	types->includes(self.activityType);


helper context bpmn!Activity def: isMessageEvent:Boolean=
	let types: Sequence(bpmn!ActivityType)=
		Sequence{#EventIntermediateMessage}in
	types->includes(self.activityType);


helper context bpmn!Activity def: isTimerEvent:Boolean=
	let types: Sequence(bpmn!ActivityType)=
		Sequence{#EventIntermediateTimer}in
	types->includes(self.activityType);



helper context bpmn!Activity def: isXorJoin:Boolean=
	self.activityType=#GatewayDataBasedExclusive and    --XOR
	self.incomingEdges.size()>self.outgoingEdges.size(); --input >
output

helper context bpmn!Activity def: isXorSplit:Boolean=
	self.activityType=#GatewayDataBasedExclusive and    --XOR
	self.incomingEdges.size()<self.outgoingEdges.size();--input <
output


helper context bpmn!Activity def: isEbXorJoin:Boolean=
	self.activityType=#GatewayEventBasedExclusive and   --EBXOR
	self.incomingEdges.size()>self.outgoingEdges.size(); --input >
output

helper context bpmn!Activity def: isEbXorSplit:Boolean=
	self.activityType=#GatewayEventBasedExclusive and   --EBXOR
	self.incomingEdges.size()<self.outgoingEdges.size();--input <
output

helper context bpmn!Activity def: isAndJoin:Boolean=
	self.activityType=#GatewayParallel and      --AND
	self.incomingEdges.size()>self.outgoingEdges.size(); --input >
output

helper context bpmn!Activity def: isAndSplit:Boolean=
	self.activityType=#GatewayParallel and      --AND
	self.incomingEdges.size()<self.outgoingEdges.size();--input <
output

helper context bpmn!Activity def: isEndEvent:Boolean=
	let types: Sequence(bpmn!ActivityType)=
		Sequence{	#EventEndEmpty,
					#EventEndMessage,
					#EventEndError,
					#EventEndCompensation,
					#EventEndTerminate,
					#EventEndLink,
					#EventEndMultiple,
```

```
                                        #EventEndCancel}in

        types->includes(self.activityType);


helper context bpmn!Activity def: isTask:Boolean=
        self.activityType = #Task;

helper context bpmn!Activity def: nodeType:babel!NodeType=
                let nodeTypes: Map(Boolean,babel!NodeType)=
                Map{
        (self.isStartEvent, #StartEvent),
        (self.isMessageEvent, #MessageEvent),
        (self.isTimerEvent, #TimerEvent),
        (self.isXorJoin, #XORJoin),
        (self.isXorSplit, #XORSplit),
        (self.isEbXorJoin, #EBXORJoin),
        (self.isEbXorSplit, #EBXORSplit),
        (self.isAndJoin, #ANDJoin),
        (self.isAndSplit, #ANDSplit),
        (self.isEndEvent, #EndEvent)

                } in
                nodeTypes.get(true);
rule Main {
        from
                d: bpmn!BpmnDiagram
        to
                r: babel!DocumentRoot(
                bpmn<- bp
                ),

                bp: babel!Bpmn(
                        process<- p
                ),
                p: babel!Process(
                        id<-d.name,
                        nodes<-nodes,
                        arcs<-arcs
                ),
                nodes: babel!Nodes(
                        node<-bpmn!Activity.allInstances() -- -
>union(bpmn!Activity.allInstances()->select(a|a.splits and a.isTask)-
>collect(x|thisModule.resolveTemp(x,'gate')))
                ),
                arcs: babel!Arcs(
                        arc<-bpmn!SequenceEdge.allInstances()-- -
>union(bpmn!Activity.allInstances()->select(a|a.splits and a.isTask)-
>collect(x|thisModule.resolveTemp(x,'arc')))
                )
}
rule Activity2Node{
        from a: bpmn!Activity
        to
        node: babel!Node(
                id<-a.iD,
                name<- a.name,
                type <- a.nodeType)
}
rule Sequence2Arc{
        from s: bpmn!SequenceEdge
        to
        arc: babel!Arc(
                id<-s.iD,
                source<- s.source.iD,
                target<- s.target.iD,
                guard<-s.name)
}
```