# Automatic execution of business process models: exploiting the full benefits of Model-Driven Engineering approaches

J. Fabra[a,*], V. De Castro[b], P. Álvarez[a], E. Marcos[b]

[a]*Department of Computer Science and Systems Engineering, University of Zaragoza, María de Luna 1, E-50018 Zaragoza, Spain*
[b]*Kybele Research Group, Rey Juan Carlos University, Tulipán S/N, E-28933 Móstoles, Madrid, Spain*

## Abstract

The business goals of an enterprise process are traced to business process models with the aim of being carried out during the execution stage. The automatic translation from these models to fully executable code which is capable of being simulated and round-trip engineered is still an open challenge in the Business Process Management field. Model Driven Engineering has proposed a set of methodologies with which to solve the existing gap between business analysts and software developers, but the expected results have not as yet been achieved. In this paper, a new approach to solve this challenge is proposed. This approach is based on the integration of SOD-M, a model-driven method for the development of service-oriented systems, and DENEB, a platform for the development and execution of flexible business processes. SOD-M provides business analysts with a methodology that can be used to transform their business goals into composition service models, a type of model that represents business processes. The use of the Eclipse Modelling Framework and the ATLAS Transformation Language allows this model to be automatically transformed into a DENEB workflow model, resulting in a business process that is coded by a class of high-level Petri-nets and is directly executable in DENEB. The application of the proposal presented herein is illustrated by means of a real system related to the management of medical images.

*Keywords:* Business Processes, Model-Driven Engineering, Model transformation, Model execution, Service-oriented development

## 1. Introduction

In recent years, the rapid development in the field of Web service technologies and Service-Oriented Computing (SOC) has motivated an interest in the area of Business Process Management (BPM) [1]. BPM has broadened to become a set of technologies and standards for the design, execution, administration

[*]Corresponding author. Tel.: +34 976 762 874 / Fax: +34 976 761 914
*Email addresses:* `jfabra@unizar.es` (J. Fabra), `valeria.decastro@urjc.es` (V. De Castro), `alvaper@unizar.es` (P. Álvarez), `esperanza.marcos@urjc.es` (E. Marcos)

and monitoring of business processes, and also as a means to deal with frequent changes in business and value chains [1, 2]. In conceptual terms, a business process is a defined set of activities that represents the steps required to achieve a business objective [3]. In BPM terminology, an activity can be seen as the work of a person, an internal system, a service provided by an external entity or the process of a partner company.

The motivation in BPM and Service Oriented Architectures (SOA) areas has also provoked the emergence of several languages for the design and implementation of such processes. The Web Services Business Process Execution Language, WS-BPEL [4], is a standard language widely used for the specification of executable business processes. However, WS-BPEL and most of the current existing approaches for the specification of processes have some relevant problems: it is difficult for business analysts to use them in the early stages of the development process [5] (analysis and modelling stages), their dependence on a specific implementation technology (more specifically, on the Web service technology) and, finally, the lack of formal semantics that permit process analysis. These problems increase the gap between business analysts and software developers, which represents a serious limitation in the BPM field.

A new beta version of the Business Process Model and Notation standard, BPMN 2.0 [3], has recently been published as a future means to reduce this gap. The informal formalization of the execution semantics for all BPMN elements and the independence of specific implementation technologies proposed by the specification will solve the first two problems associated with previous approaches. Nevertheless, this promising proposal lacks formal semantics and needs time to mature. The aforementioned gap is, therefore, still open.

In order to overcome these limitations, the Web Engineering field has proposed a set of methodologies which make it easier to develop service-oriented systems based on current technologies. Model Driven Development (MDD) is a development methodology which is principally characterised by the use of models as a product [6]. MDD is a subset of Model Driven Engineering (MDE) [7] because, as the E in MDE suggests, MDE goes beyond pure development activities and encompasses the other model-based tasks of a complete software engineering process [8]. Model Driven Architecture (MDA) [9] is the particular MDD methodology defined by the Object Management Group (OMG) and therefore relies on the use of OMG standards. More specifically, MDA offers an open, vendor-neutral approach with which to master technological changes and provides a conceptual structure for: i) specifying a system independently of the platform that supports it; ii) specifying software platforms; iii) choosing a particular platform for the system; and iv) transforming the system specification into the corresponding code for a particular target platform (that is, moving the system specification to the technological one).

Two fundamental ideas are behind the Model-Driven Development of software systems. First, MDD methodologies provide a set of models with which to cover the different stages involved in the design of a system. These models are consistent with each other and include traceability between different abstraction levels of the design. Second, the models are explicitly separated from implementation technologies and execution platform details. This last characteristic allows developers to reuse software components and thereby avoids the need for repeated investments in time and money. Both ideas have been used to define Model-Driven Architecture (MDA) approaches which can specify the functionality of each type of software systems [10]: MDA for service-oriented information systems, for business processes, for embedded systems, etc.

However, after several years of the popularization of MDA proposals and the appearance of many MDD approaches for software development, the usefulness of these approaches is questionable [6]. The main controversy is related to the lack of support in the automatic generation of code from models. Ideally, technological frameworks should be able to transform models into executable code and then execute the resulting code without human intervention [11]. This ability has an influence on several stages of the lifecycle of a software system, principally with regard to its development and in the analysis prior to its deployment. Automatic code generation has traditionally been restricted to the generation of code skeletons and fragments, requiring human intervention to complete them. In order to overcome the limitation imposed by this kind of automatization, modeling and implementation languages should converge to facilitate the generation of complete programs. Moreover, the code generated could also be used for the automatic verification of models on a computer. This signifies checking that the requirements fulfill the business goals and that there are no undesirable properties in the models. From an empirical point of view, model verification can be based on the simulation of generated programs [6]. Nevertheless, more powerful and formal analysis methods and techniques should be integrated into MDA approaches to verify a system's model (model checking or logical inference, for instance). Code generation therefore emerges as a key element if the usefulness of MDA solutions is to be achieved in different stages of the development of a software system.

This paper presents the integration of the Service-Oriented Development Method, SOD-M [12], and the platform for the Development and Execution of iNteroperable dynamic wEB processes, DENEB [13]. From the point of view of MDD, SOD-M defines a model-driven method for the development of service-oriented information systems. In this method, the key concept is the use of composition service models (which are equivalent to business process models) to represent the Platform-Independent Model (PIM) level of a system. On the other hand, from an execution point of view, DENEB provides an environment for the deployment and execution of business processes. Our integration proposal therefore consists of automatically translating SOD-M composition models into business processes that are directly executable by DENEB. This solution solves the gap between business analysts (their business goals have been represented in SOD-M models) and software developers (their implementations of models are automatically generated). With regard to existing approaches our proposal presents three relevant contributions in connection with the resulting business processes: firstly, they are completely generated without human intervention; secondly, they can integrate a wide variety of implementation technologies (not only Web services); and, finally, they have formal semantics (a class of high-level Petri-nets has been used to code DENEB processes), and could therefore be analyzed by using formal analysis techniques [14, 15]. From a technological point of view, both the model transformation and the code generation utilities required by our proposal have been developed using the results of projects supported by Eclipse [16], the ATLAS Transformation Language (ATL) [17] and the Eclipse Modeling Framework (EMF) [18].

The remainder of this paper is structured as follows. Section 2 presents the main work related to the BPM modeling area, the execution of business processes and code generation from business process models. In Section 3, the foundations of the two approaches to be unified, SOD-M and DENEB, are introduced. The proposed model transformation process, the metamodels involved and the plug-in implementations are then sketched in Section 4, in which the process of moving from the composition service model in SOD-M to the workflow model in DENEB is depicted by first presenting the model implementation for both the SOD-M and DENEB approaches. The EMF pluging implementation, the model transformation and the mapping implementation using Ecore and ATL are then described. The application of the proposal is illustrated in Section 5 by means of a real use case related to the management of medical images. Finally, Section 6 concludes the paper and shows the directions of our future work.

## 2. Background

In this section the role of MDA in the BPM-SOA combination is presented, and the key points in which MDA helps and improves in the whole process are described. Several BPM standards have been proposed for their application to the different levels of MDA in order to enhance its benefits. The most relevant ones, and the most important transformation approaches to generate executable business process models, are also depicted here.

### *2.1. The role of MDA in the BPM and SOA combination*

Service-oriented development has recently become one of the major research topics in the field of software engineering, which has even led to the appearance of a new and emerging discipline called Service Engineering (SE). Service Engineering aims to bring together the benefits of SOA and BPM initiatives. The BPM and SOA combination has been proposed as the best approach by which to achieve a closed alignment between business processes and IT resources. This combination allows the rapid development of agile processes which can respond to changes in business requirements.

However, BPM and SOA represent two different initiatives. BPM is mainly a management discipline and strategy which endorses the idea that a business can be modelled in terms of its end-to-end processes which cut across organizational and system boundaries. These processes are then represented in a manner that computers can understand and process [19, 20]. On the other hand, SOA looks for a better business process alignment with service protocols, legacy applications and software components, this is, it comprises an organizational paradigm which is supported by the underlying IT infrastructure. Moreover, BPM is a business-driven, project-oriented and top-down approach, whereas SOA uses an IT-driven, enterprise infrastructure-oriented one. SOA can also be a top-down (when based on portfolio management and service analysis and design) or a bottom-up approach [21]. Finally, BPM reuses process models, whereas SOA reuses service implementations, being more dependent on technological issues.

Fortunately, BPM and SOA have evolved over time to be applied together, becoming two sides of the same coin [20]. SOA assists in BPM proliferation, allowing processes modelled using BPM tools to be implemented quickly using the loosely coupled and agile infrastructure of SOA. On the other hand, BPM provides SOA with a wide and strong business-case, facilitating the closed alignment between business and IT. The combination of BPM and SOA would reduce the investment, development and maintenance costs, since both initiatives encourage the loose coupling and spreading of internal and external applications across a distributed technology platform [11].

The OMG group has recently proposed the use of its MDA architecture as a means to model processes and services based on a platform-independent approach [22, 9]. According to Watson in [1], the convergence of SE with MDE represents one of the most important frameworks to hold out the promise of rapid and accurate development of software that serves software users' goals. By using MDA, complete SOA solutions can be developed through models, thus avoiding investment in specific technologies and protocols [23]. The Service Oriented Architecture SIG group (SOA ABSIG) coordinates the SOA standardization efforts between the OMG and other SOA standards groups such as the W3C, OASIS or the Open Group [24]. Its aims are to promote a faster adoption of SOA-specific modelling approaches and best practices and to support an MDA approach to SOA that links architectural, business and technology views of services.

### *2.2. BPM standards applied to MDA*

MDA proposes the use of different models to represent the different points of view of a system, from the higher abstraction level to the concrete technological level, namely: Computation Independent Model level (CIM), Platform Independent Model (PIM) level and Platform Specific Model (PSM) level [9]. In the specific case of BPM, several business process-related language proposals have been proposed in relation to the different MDA stages [25, 26].

At the top level (CIM level), the Business Process Modelling Notation (BPMN) standard is generally used [27]. BPMN became the predominant notation standard with which graphically depict process models thanks to the merging of the BPM initiative into the OMG group. This merging came about as a result of the overlapping scope of BPMN and UML activity diagrams. BPMN provides a graphical notation that is easy for all the participants in the business process, from business analysts to developers, to understand. It specifies business process diagrams by means of flows, events, activities and results. Business decisions and forks can also be modelled by means of gateways. BPMN activities can be elementary taks or sub-processes. BPMN also allows the participants involved in the process to be modelled by means of *pools*. In these pools, activities can be organised and categorised by means of *lanes*, although the use of these elements is left to the modeller's discretion. Several approaches have proposed the application of certain transformation techniques in order to obtain a UML equivalent model from a BPMN model and the corresponding code which that model implements [28].

The BPMN specification has recently evolved from version 1.2 (betwen 2008 and 2009) to the latest version, BPMN 2.0 (2010) [3]. This new version contains many significant changes, principally regarding execution semantics and choreography, and also important modifications in conformance, swimlanes, dataobjects, sub-processes and events. The specification has been rebuilt and now differentiates process modeling, process execution, BPEL process execution and choreography modeling. An extensibility mechanism has been defined for both process model extensions and graphical extensions. Anothe major innovation is that it extends the definition of human interactions. BPMN 2.0 therefore represents one of the most serious and suitable candidates for model description at CIM level.

Finally, the ARIS language is also applied to the CIM level, providing a graphical notation that is supported by the ARIS Toolset [29].

At the PIM level, the Business Process Definition Metamodel (BPDM) provides a graphical notation when used in conjunction with UML [25]. BPDM has been proposed as a standard by the OMG. The J2EE SUN standard, which represents the basis of most Web applications, can also be used to obtain a graphical notation when used in conjunction with UML [30].

The Business Process Modelling Language (BPML) [31], the Electronic Business using eXtensible Markup Language (commonly known as e-business XML, or formally ebXML) [32], WS-BPEL [4] and the Web Services Choreography Description Language (WS-CDL) [25] are normally used to represent the PSM abstraction level. The popularity of BPML has been outperformed with the widespread adoption of XPDL and BPMD as interchange and serialization formats [5].

SOA foundation standards are required to achieve a model-driven development in BPM tools. Without them, BPM would become slow and expensive. Several languages for the design and execution of business processes based on Web services have appeared. Some examples are XLANG [33], the Web Services Flow Language (WSFL) [34] or WS-BPEL [4]. More specifically, WS-BPEL is recognized as one of the most common SOA orchestration languages. WS-BPEL is used to describe the execution logic of Web services by defining their control flow and providing a means to share the context among the different architectures partners. The use of WS-BPEL signifies that those business processes which integrate a Web service collection in the same process flow can be constructed. The result is an interoperable, robust and standard-based SOA solution. WS-BPEL is based on several XML-based specifications such as WSDL 1.1, XML Schema 1.0, XPath 1.0 and XSLT 1.0 [4]. WSDL messages and XML Schemas provide the data model used by the processes described using WS-BPEL, whereas Xpath and XSLT support data management.

WS-BPEL also provides several extension mechanisms in order to easily adapt new versions of these standards, principally those related to XPath and XML-based standards. Curiously, several vendors do not use WS-BPEL as an execution language, but as an import/export interchange language [35].

WS-BPEL and BPMN 1.2 are sometimes thought to be similar. Nevertheless, BPMN 1.2 is used in a different development stage to WS-BPEL, specifically when the business process is being designed and improved, whereas BPEL is used to implement it. Obviously, these two phases have completely different requirements [36]. Moreover, BPMN 1.2 is used by business analysts and WS-BPEL is used by technical analysts and programmers. Each of them uses different paradigms and is focused on different aspects and issues when modelling a process. BPMN 1.2 and WS-BPEL were not originally intended to work together. From a standardization point of view, this represents an important gap which should be solved before processes can be modelled and deployed within a BPM-MDA unified framework.

However, the publication of BPMN 2.0 has changed this differentiation. Moreover, BPMN 2.0 is intended to replace WS-BPEL, although some steps must be taken in this area if such a task is to be achieved. BPMN 2.0 still has some issues, such as semantics of non-executable models, model portability, graphics interchange or simulation properties [37]. There are also some problems with support to prior versions, and the validation tools are still rather inmature. The portability of BPMN 2.0 process models is a desirable and achievable goal which business analysts have expected. However, the existence of multiple schemas and separate graphics complicates validation, and the graphic portability in existing tools is becoming more difficult. BPMN 2.0 plus WS-BPEL does not therefore solve the portability problem for business analysts, and there are still some differences between both standards. However, it is clear that in recent years the evolution of BPMN 2.0 has overcome the limitations and differences of its previous version as compared with WS-BPEL.

### *2.3. Generating executable business processes from models*

A key aspect in a standard-based description of a business process is the generation of source code to implement it in a semi-automatic manner, that is, with minimum human intervention. This would allow business analysts to generate a process model which could then be translated into an implementation language. At this stage, software developers may be able to complete the process implementation being compliant with the analysts' requirements. The use of BPM standards aims to ease this process. However, the main weakness of most of the BPM standards presented is the difficulty involved in their use by business analysts during the early stages of the development process [5]. In most cases, these standards only address those processes that are composed exclusively of Web services; they do not have a graphical rendering; they do not usually include a framework to perform a top-down design and they do not provide capabilities for process analysis. Therefore, the transformation from the high-level business process modelling, which is generally only carried out by business analysts, into a composition language which implements those business processes with Web services is not a trivial matter.

In [38], the authors provide a set of rules with which to generate the corresponding code between a BPMN 1.1 process model and its corresponding WS-BPEL executable process. To do this, the object diagrams and their properties are dissected and then mapped into the appropriate WS-BPEL elements. A BPMN 1.1 diagram therefore attains gets a double utility/finality: to provide the business process from the perspective of the business-level view, and to allow executable code to be generated in WS-BPEL. This work describes the transformation process by means of examples, making a distinction between graph structures (*flow element*) and WS-BPEL blocks (*sequence element*). However, it does not provide any methodology or implementation to perform this transformation in an automatic manner. Indeed, the features of a tool able to model an annotated BPMN 1.x process with the information required to perform the transformation are depicted, but the tool is not developed. The transformation of BPMN 1.x into BPEL is also carried out using the Oracle Business Process Analysis Suite in [36]. This tool is complementary to the Oracle Business Process Management Suite, and can be used to formally model the business architecture of an organization using the BPMN notation. The differences between both BPMN 1.x and WS-BPEL are identified and analysed, and a transformation process is depicted. However, when using this method some semantic differences exist between BPMN 1.x and the final WS-BPEL models. The use of this proposal should therefore be limited to technical and simple models (with a few services) and it could not be applied to more complex requirement scenarios. With regard to BPMN 2.0, the interesting work presented in [39] depicts the development of a precise execution semantics for a subset of BPMN 2.0 using graph rewrite rules. The justification of using these rules is the strong relation between the execution semantic rules which are informally specified in the 2.0 standard and their formalization. The authors propose two different transformations. The first is from BPMN 2.0 to the workflow system YAWL [40], thus allowing models to be executed in workflows. The other is to the Petri net formalism [41], which allow BPMN 2.0 models to be formally analysed .

The XML Process Definition Language, XPDL, defines an XML schema to specify the declarative part of workflow / business process [42]. XPDL permits the graphic representation of business processes, specifying attributes such as roles, activity description, timers, Web service invocations, etc. The aim of XPDL is to store and allow the exchange of BPMN process diagrams. It offers a standard means to represent processes and to import/export them to/from any editor that is compliant with the standard. XPDL 2.0 has the extensions needed to represent all the aspects of BPMN. In [43], an interesting approach that integrates BPMN 1.0, XPDL and WS-BPEL is presented. The starting point is a a BPMN 1.0 model, and the partial diagrams are stored as XPDL during the design stage (in the case of XPDL 2.0 this is immediate). These parts are then translated into WS-BPEL, and a complete transformation process is provided. However, this process is limited to data exchange, since human interactions cannot be modelled in WS-BPEL. This task cannot be carried out without the use of specific and vendor-dependant modules.

The WebML approach offers a Web-based perspective and allows business processes to be specified according to the BPMN notation, and these are then transformed from the BPMN notation to WebML hypertext diagrams [44]. These diagrams then allow the fast generation of skeletons which implement the specified business process. However, these skeletons are not suitable for the implementation of the whole process, and a developer must therefore complete the code with the remainder and specific implementation details.

Some other proposals are based on UML, such as [44] and [30], and others define their own models, as in [12] and [11]. The main advantage of these proposals is that they are generally supported by modelling tools which include modules for automatic code generation from those models to Web Service composition languages (for example, OptimalJ and ArcStyler [45]). However, these proposals contemplate isolated models that cannot be easily and automatically mapped on a final target platform. The proposal used in this work allows obtaining a service composition model that can be almost directly or automaticly mapped onto executable languages.

The UML 2.0 modelling activity notation has been widely criticized because of its similarities to the BPMN notation [46]. Although UML covers activity modelling, class modelling, state machine modelling and more, a model of a business process described using UML is not a model of software, but is still a model of a business process. However, the Executable UML approach (also called xtUML or xUML) supports the specification of PIM models and their compilation into PSMs [47]. Other approaches have also proposed the generation of XPDL specifications from UML activity diagrams [48, 49, 50].

Finally, the OMG has collected a comprehensive list of successful histories (vendors and products) when performing direct transformations from PIM to PSM models [51].

As shown in this section, MDA serves as a bridge between BPM and SOA. The use of MDA allows processes and services to be modelled by following a model-based approach, along with the generation of platform-independent process models. The use of BPM standards when applied to MDA also helps in the business process alignment when developing a SOA solution. However, it is clear that the use of these standards creates a serious gap in the development stages. These standards are principally difficult for business analysts to be used in the early stages of the development process. Moreover, several limitations exist in the semi-automatic generation of executable code for the process descriptions provided. These limitations are generally related to the generation of code skeletons which must be completed by software engineers in later stages. A few approaches allow a complete process implementation to be generated. However, these approaches normally produce solutions that are composed exclusively of Web services, which is not normally the business analyst's aim, as occurs for example in the case of WS-BPEL.

## 3. Underlying Approaches

This section introduces both the SOD-M and DENEB approaches integrated into this work and depicts the key concepts behind their integration.

### *3.1. SOD-M: Service Oriented Development Method*

SOD-M defines a service-oriented approach for the development of information systems (IS) [12, 52]. Its main characteristics are the provision of a set of guidelines with which to build IS based exclusively on services. SOD-M proposes the use of services as first-class objects for the whole process of the IS development. This approach facilitates the development of service-oriented applications, along with their implementation using current technologies such as Web services.

The method provided in SOD-M follows an MDA-based approach. It proposes a set of models which cover from the highest level of abstraction of the MDA, the CIM level, to the PIM and PSM levels. SOD-M provides the benefits related to the alignment of high-level business processes with the application of current technologies to deploy the Service-Oriented paradigm by means of different mapping rules between the models [23].

SOD-M focuses on the development of *behavioral aspects* of IS and defines the guidelines needed to build behavioral models from high-level business modelling. The method is part of a larger MDA framework called MIDAS [53] in which other aspects such as *content* [54] and *hypertext* [55] are considered. Content models are thus derived from a domain model defined at CIM level, and hypertext models are derived by taking into account the content models (which contain data that need to be shown) and the behavioral models (which contain the steps and functionalities needed to carry out specifics tasks).

The concepts used in SOD-M to model the behaviour of an IS are organized from two different points of view. On the one hand, the business view focuses on the features and requirements of the business in which the IS will be built. On the other hand, the information system view focuses on the functionalities and processes which need to be implemented in the IS in order to satisfy the business requirements.

Figure 1 depicts the modelling process of SOD-M. It includes models that correspond with: i) the three different abstraction levels considered by MDA: CIM, PIM and PSM; and ii) SOD-M views: business and information system views.

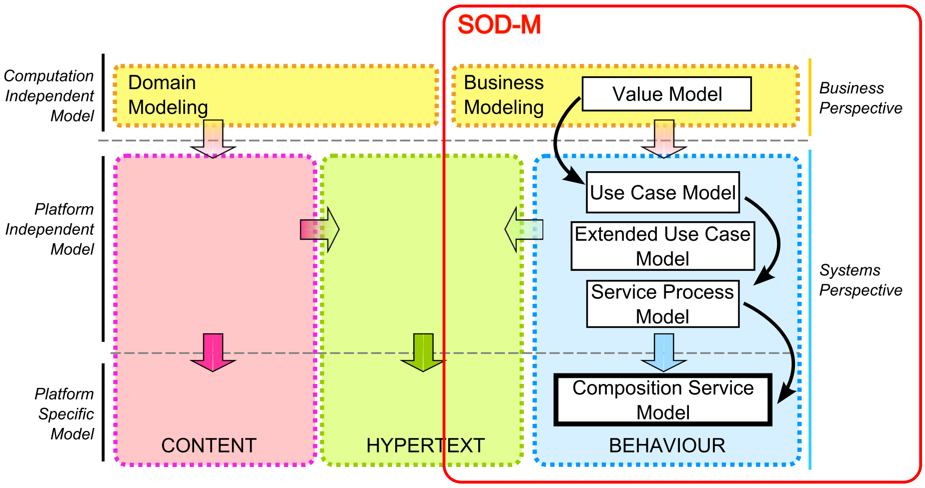


Figure 1: SOD-M development process.

As is shown in Figure 1, the SOD-M model-driven process begins by building the high-level computational independent models and enables specific models for a service platform to be obtained as a result.

For the sake of simplicity, in this work we shall focus on the *composition service model* of SOD-M (highlighted in Figure 1). The *composition service model* is the model which will be taken as input for the transformation process to generate the corresponding workflow model that will be executed in the DENEB platform. Note that SOD-M proposes representing this model by using the UML activity diagram technique [46]. The main reason for using the UML activity diagram notation and not BPMN is because the latter is basically a software development method, whereas UML is a standard notation that is well known by software developers. As is argued in [56], in which BPMN and UML activity diagrams are compared in terms of workflow patterns, although both notations are very similar and represent views of the same metamodel, the main differences between the two diagrams consist of the users' targets. The BPMN notation was created to be used by business people, while the UML activity diagram is a standard notation for software (IS) development. A more detailed description of the other different SOD-M models can be found in [12]. A more detailed description of the service composition model, its elements and the metamodel implementation using Eclipse will be presented in Section 4.1.

### *3.2. DENEB: business processes based on the conversational approach*

DENEB is an open, flexible, extensible and adaptive framework for the development and execution of interoperable dynamic Web processes [57, 13]. In

DENEB, processes are composed of a set of tasks to be executed, and these tasks are arranged according to a specified set of ordering constraints. DENEB processes are described and executed using workflow technology, in terms of which is usually called *business logic*. On the other hand, the execution of each task usually corresponds to the invocation of either a particular internal process or published service. The specification of the correct sequences of messages exchanged among communicating entities is referred to as an *interaction protocol*. In DENEB, these protocols are organized as a set of roles. A *role* is the part of the protocol that a participant process or service must execute. In an interaction protocol, many alternative and valid sequences of messages often exist, each of which triggers one action or another. The execution of one of these possible sequences is called a *conversation*.

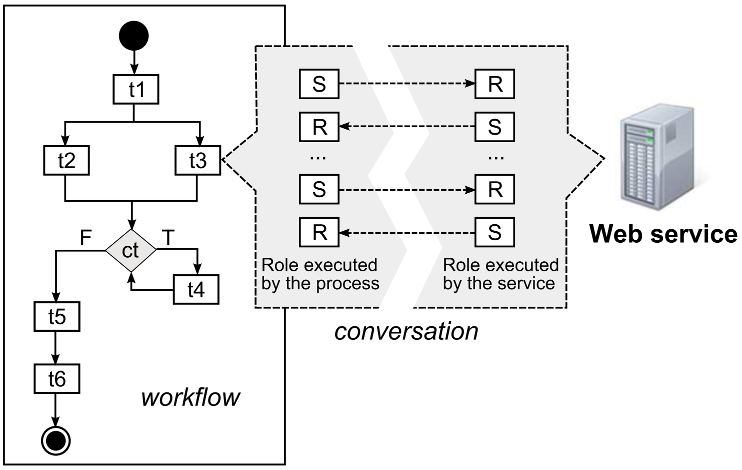


Figure 2: Abstraction of a business process executed by DENEB.

Figure 2 shows an abstraction of a business process executed by DENEB. The process consists of a workflow which implements its business logic. If a task needs to interact with an external service (let us say task *t3*, for instance), the process must execute the required role in order to interact with the selected service. A service registry may be used beforehand to discover the role specification. The sending and receiving of messages (denoted by $S$ and $R$, respectively) by the process during the role execution are encapsulated as a part of the task. Nevertheless, a clear and explicit separation between the business logic aspects (that is, the workflow) and the interaction aspects (that is, the roles) is kept in order to achieve flexible service integration. Consequently, the same workflow could execute the roles of alternative candidate services for each specific task. This methodology for the design of business processes is called *the conversational approach* [58].

Both the workflows and roles executed in DENEB are implemented using Reference nets, a subclass of high-level Petri nets belonging to the *Nets-within-Nets* paradigm in which tokens in the system net can be references to other nets, thus making it possible for different tokens to point to the same instance [59]. Reference nets are described internally by means of the Petri Net Markup Language, PNML, a XML-based standard language for the interchange of Petri nets, and they are directly executable by means of the Renew tool [60, 61]. DENEB also allows PNML descriptions of both workflows and protocols to be imported, and their acquisition at runtime [13].

### *3.3. The operating environment of DENEB*

DENEB has been built as a SOA-based platform that is able to execute business processes which are compliant with the previous description. The architectural model of DENEB uncouples business logic aspects from interaction logic aspects (workflows and their related protocol roles, respectively).

Figure 3 depicts an overview of the DENEB architecture. Note that these architectural guidelines correspond with the requirements of the conversational approach. The core of the platform consists of three main components: a *workflow engine*, a *conversation engine* and a *message broker*. The workflow engine executes the business logic of processes, whereas the conversation engine executes their corresponding interaction roles. Both engines communicate in order to exchange information between a workflow and its roles (such as parameters for the service invocation or results from external invocations). The message broker is responsible for supporting this communication and coordinating interactions by means of an internal message repository in which normal messages or control/synchronization messages can be stored. From a design point of view, the broker infrastructure facilitates the independence and decoupling in time of the engines. Different implementations of engines could be plugged into the broker infrastructure and they would cooperate by means of the exchange messages using a common and defined format [62].

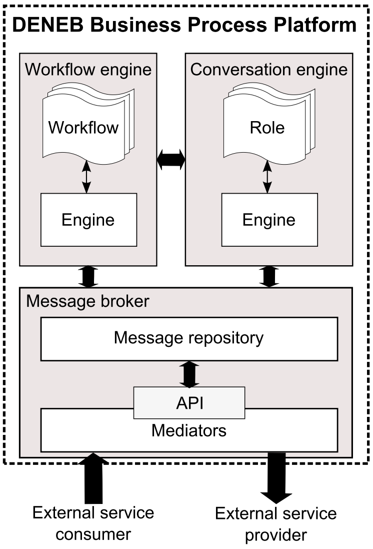


Figure 3: SOA-based architecture of DENEB.

Additionally, the message broker is responsible for the management of communication with external services. *Mediators* provide all the means needed to send and receive messages using different transport protocols and message formats. They also serve to isolate the business process model from the details of particular protocols, such as SOAP, RPC, HTTP, SMTP, etc., thus enabling communication with external entities which use a wide variety of communication technologies. Internally, the roles are executed by the conversation engine, and mediators communicate and are accessed through the message repository. When a role needs to communicate with an external service, it stores a tagged message in the repository. This means that this message will be taken and processed by the corresponding mediator, which will interact with the target service. Service responses are similarly received and processed by mediators and then delivered to the corresponding role through the repository.

Mediators represent a powerful tool with which to integrate new features, capabilities and functionalities into DENEB in a non-intrusive manner (that is, without having to modify the core of the platform). The only requirement for adding a new mediator is that it must be compliant with a predefined API. Some examples of mediators used in DENEB are an encapsulated business rule engine (BRE) to extend the executing platform with decision-making features, a security component to allow secure communications using a PKI schema, or a new component for supporting human-user interactions during the execution of business processes.

Finally, from an implementation point of view, Reference nets [59] were used to implement the workflow and conversational engines, along with a distributed message broker based on the Linda coordination model [63], called DRLinda [62]. The lifecycle of business processes (deployment, initiation, execution, interaction and termination), their corresponding conversations and the capabilities and functionalities described above are also managed using Reference nets. We should stress that the same formalism has been used to design/implement the business processes and its runtime environment: the DENEB platform itself.

### *3.4. Why to integrate DENEB with SOD-M?*

SOD-M and DENEB provide several individual contributions on different levels of the software development lifecycle. On the one hand, SOD-M addresses the main problem of typical languages for business process specification: their limitations when being used in the early stages of the software development process by business analysts. SOD-M provides a top-down design framework along with capabilities for the process analysis. Its method provides a composition service model, starting from the user requirements, which specifies the services required to achieve a common business goal. On the other hand, DENEB provides an infrastructure for the execution of processes composed of services and different interaction technologies; it has a graphical rendering interface; and it provides a flexible and easily extensible mechanism to support the dynamic capabilities of processes [13]. DENEB uses the formalism of the Nets-within-Nets which facilitates the representation of executable workflows and protocols by following the conversational approach. This design approach improves the flexibility in the development of processes and decouples business and interaction logic.

The objective of our proposal is to integrate both existing approaches in order to provide a complete framework for the analysis, design, development and execution of business processes. This integration is based on the ability to generate executable code (more specifically, executable DENEB processes) from SOD-M composition service models. Compared to existing approaches, this represents an innovative integration work, since the resulting processes are generated without human intervention, they are fully and directly executable, and they have formal semantics that can be used for their verification (as will be detailed in Section 6). The transformation from high-level business process modeling, which is generally carried out by business analysts, to a composition language which implements these business processes is not a trivial matter. However, as is shown in this work, this process can be achieved more easily by means of model transformations. EMF plugin implementation, model transformations and mapping implementation by means of Ecore and ATL are used to integrate the SOD-M and DENEB approaches, and this is shown to be a very suitable approach.

Furthermore, existing approaches are based on standards for the specification and implementation of business processes. These standards and Web service technologies are strongly connected (WS-BPEL is a good example of this connection). Regarding this issue, our proposal for integration is more flexible than existing approaches. The use of DENEB makes it possible for executable processes to communicate not only with Web services, but also with any external entity that is accessible via the network (a GRID system, for instance). What is more, from the point of view of SOD-M, these interactions could be easily modeled using executable activities in service composition models. Our integration therefore results in a unique, extensible and flexible framework which covers all development stages.

## 4. Model Transformation Process

In the scope of MDE, the model transformation process essentially targets the generation of a *DENEB workflow model* starting from a *composition service model*. However, it is important to stress that the composition service model is not modelled from scratch at this level, but is obtained as a result of a general process defined by SOD-M in which a set of models are built following a Service-Oriented approach, as was depicted in the previous section.

Let us first introduce some of the facilities that are involved in our model transformation process. *Models* represent the basic unit of construction of MDA. Models are defined according to the semantics of *a model of models*, also called a *metamodel*. Once the metamodel has been defined, any model must *conform* to this metamodel, signifying that it respects its semantics. Several technologies are available for the definition of metamodels.

The Eclipse Modelling Framework (EMF) [64] has been used to carry out the entire model transformation process. The EMF project is a modelling framework and code generation facility which is used to build tools and other applications based on a structured data model. EMF starts from a model specification described in XMI, and provides developers with tools and also runtime support to produce a set of Java classes for the model along with a set of adapter classes which enable viewing and command-based editing of the model, in addition to a basic editor.

The transformation process was automated by using ATL to carry out the transformation between entities from both the SOD-M and DENEB metamodels. ATL is a model transformation language and toolkit developed by the ATLAS Group (INRIA & LINA) [17]. An ATL program is basically a set of rules that define how source model elements are matched and navigated to create and initialize the elements of the target model. ATL uses EMF to handle models: to serialize and desterilize them, and to navigate and to modify them. ATL therefore works perfectly with the models defined using EMF editors.

Let us now focus on the model transformation process, for which Figure 4 sketches a general overview. Both the metamodel of the composition service and the metamodel of the workflow have been developed using EMF (`CompServiceMetamodel.ecore` and `WorkflowMetamodel.ecore`, respectively). The editor needed to support the graphical representation of composition service models has been obtained from its metamodel using GMF. The mapping plug-in needed to carry out the mappings between both SOD-M and DENEB metamodels is constructed using ATL transformation rules (*CompService2Workflow* plug-in). In this plug-in, *CompositionServiceModel2WorkflowModel* implements the designed transformation, which is expressed by means of the ATL language and conforms to the ATL metamodel. As has previously been stated, the three metamodels (*Composition Service Metamodel*, *Workflow Metamodel* and *ATL* itself) were expressed using the semantics of the Ecore metametamodel.

As a result of the mapping process, an instance of a workflow model and its related protocol models are then obtained starting form a particular composition service model. Both the resulting workflow and the protocols are coded using DENEB high-level Petri nets. The workflow represents the business logic modelled in the composition service model, whereas protocols implement the interaction logic (which is normally linked to service invocations and interactions). The resulting net instances are then directly executable in the DENEB platform.

### *4.1. Model implementation*

Figure 5 depicts the composition service metamodel proposed by SOD-M (at the top) and the conformance relation with regard to a composition service model (at the bottom). Although this model will be detailed in Section 5 when the use case is introduced, it is shown here because it comprises most of the main components of the metamodel. The composition service model represents a workflow needed to carry out a business service, identifying those entities that collaborate in the business processes and the actions that each of them performs. As was previously stated, this model is represented using the UML activity diagram technique. Thus, as can be seen in Figure 5, the metamodel includes typical modeling elements of the activity diagram such as *ActivityNodes*, *InitialNodes* and *FinalNodes*, *DecisionNodes*, etc., along with new elements defined by SOD-M such as *Collaborators*, *ServiceActivities* and *Actions*. Some of the meta relations between the metamodel and the depicted model are represented by the highlighted dashed boxes and arrows in Figure 5. Note that for the sake of clarity, only some, and not all of the metarelations are shown.

Let us now describe the main elements of the composition service model. *Collaborator* elements represent those entities that collaborate in the business processes by performing some of the required actions, and are displayed as a partition in the activity diagram. A collaborator can be either internal or external to the system being modelled. When the collaborator of the business is external to the system, the attribute `IsExternal` of the collaborator is set to *true*.

*Actions*, a kind of *ExecutableNode*, are represented in the model as an activity. Each *Action* identified in the model describes a fundamental behaviour unit which represents some type of transformation or processing in the system

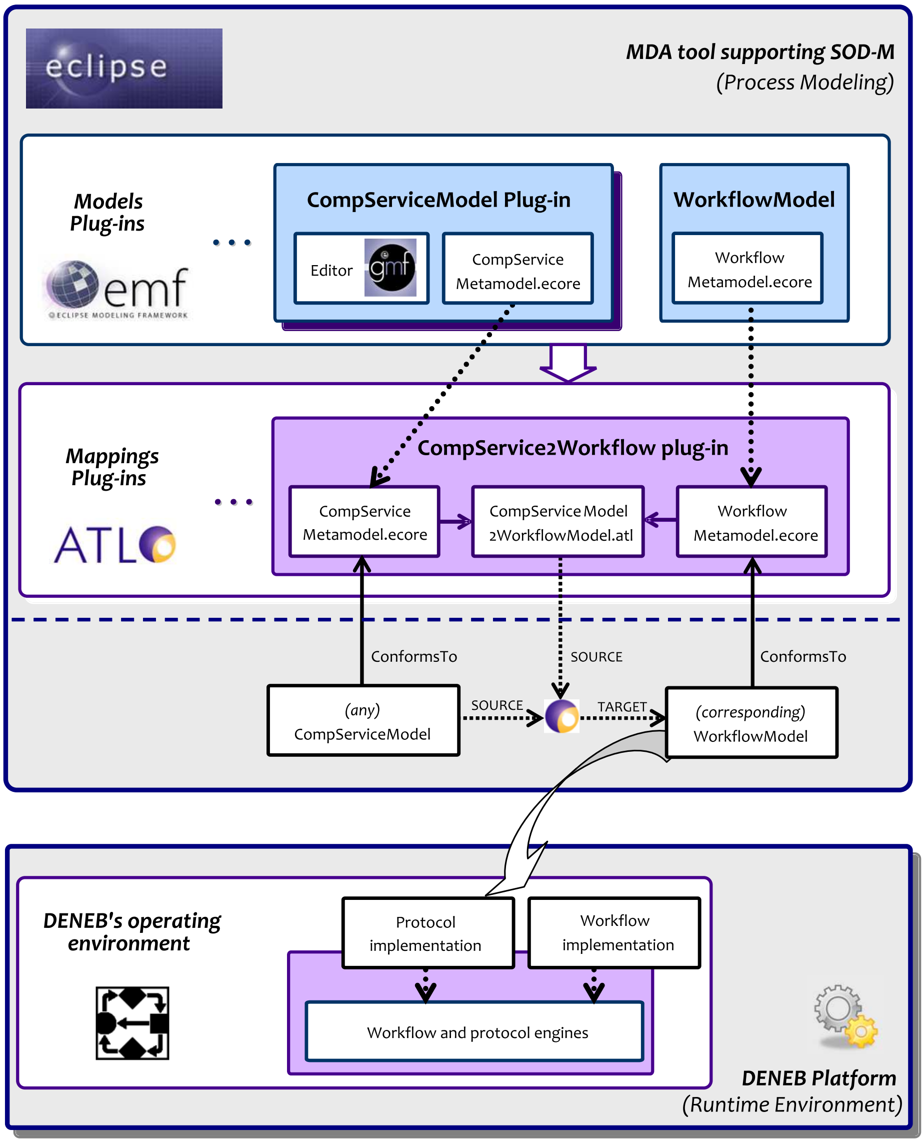


Figure 4: General overview of the model transformation process.

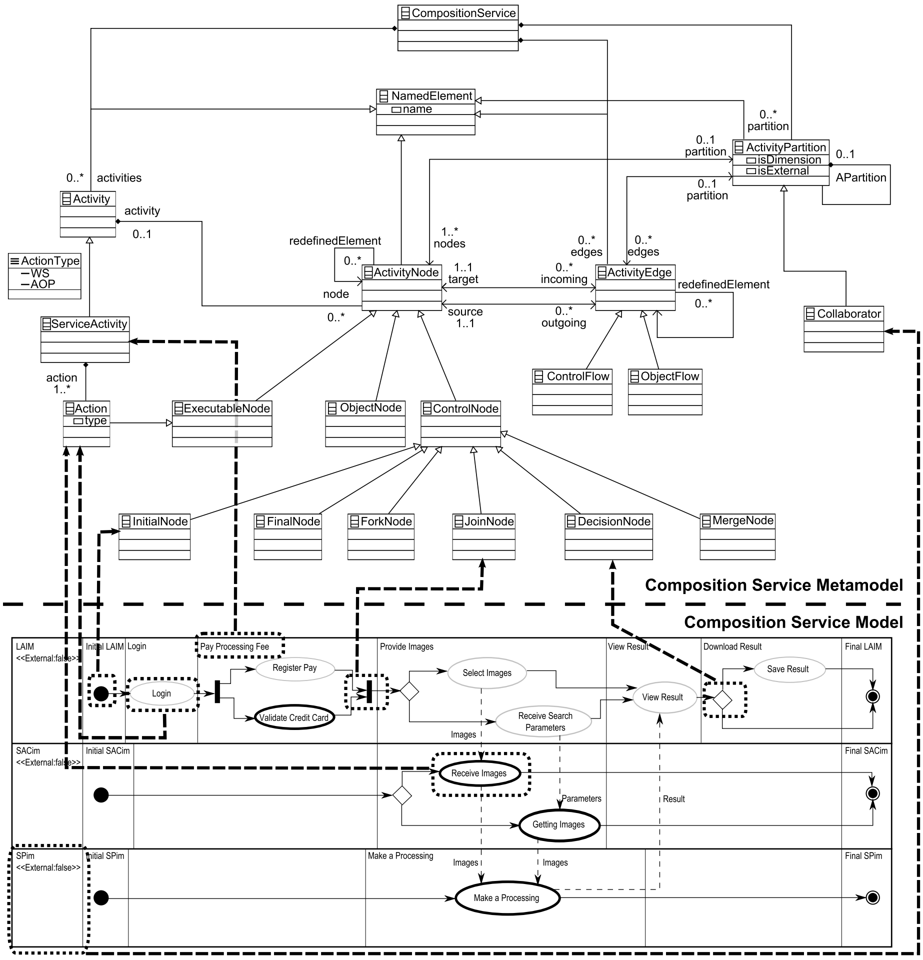
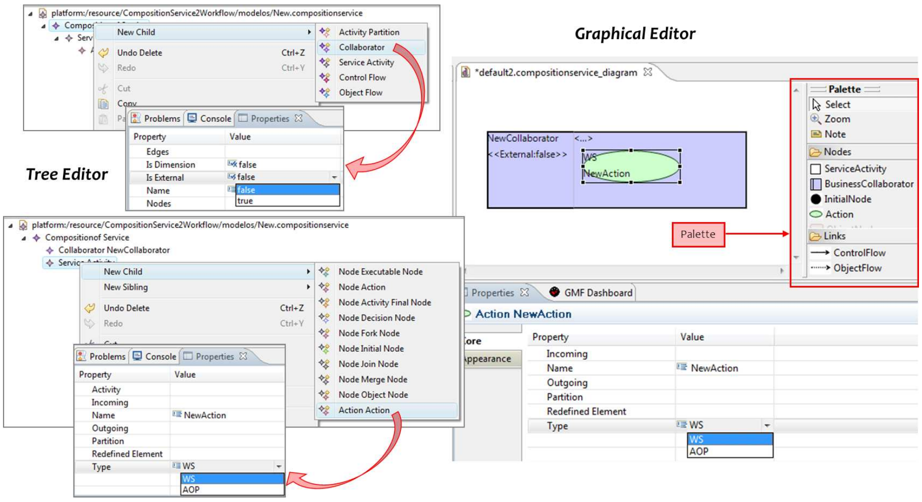


Figure 5: Conformance and meta relations with regard to the composition service metamodel.

being modelled. There are two types of action: i) a *WebService* (attribute `Type` is *WS*); and ii) a simple operation that is not supported by a Web Service, called an *ActivityOperation* (attribute `Type` is *AOP*). The *WebService* element (highlighted in bold type in Figure 5) specifies those activities which are or will be supported by Web services, depending upon whether they are implemented or not. A *WebService* element has as a tagged value which contains the set of operations it performs and the information required to achieve the Web service invocation on a concrete platform (endpoint, operation name, input and output parameters). The *ServiceActivity* element is an activity that must be carried out as part of a business service and is composed of one or more executable nodes.

The editor of the composition service model represents a very important aspect in easing the task of handling and building instances of models. It was implemented by performing the following tasks. First, the metamodel of the model was developed using EMF; EMF was then used to generate a tree-editor for the representation of models; finally, the graphical editor was developed using the Graphical Modelling Framework (GMF) in order to enable the models to be graphically depicted in a UML-like manner. Figure 6 shows a screenshot of the tree-view (left-hand side) and the graphical-view (right-hand side). The tree-editor allows an instance of a Composition Service model to be built following a tree-based hierarchical methodology. This is sometimes more useful for testing and debugging, or simply to set specific parameters. Furthermore, the graphical editor provides developers with a palette to facilitate the construction of instances. Note how certain internal details, such as the specification of the type of action depicted in Figure 6, can easily be established as WS or AOP in both views.



| Property | Value |
|---|---|
| Edges | |
| Is Dimension | false |
| Is External | false |
| Name | false<br>true |
| Nodes | |

| Property | Value |
|---|---|
| Activity | |
| Incoming | |
| Name | NewAction |
| Outgoing | |
| Partition | |
| Redefined Element | |
| Type | WS<br>WS<br>AOP |

| Property | Value |
|---|---|
| Incoming | |
| Name | NewAction |
| Outgoing | |
| Partition | |
| Redefined Element | |
| Type | WS<br>WS<br>AOP |

Figure 6: Partial view of the composition service model editor.

Figure 7 depicts the DENEB's workflow metamodel and the conformance relation with a simple workflow model. This model depicts a simple task for which a new conversation is instantiated. Some data is exchanged with it by means of the processing of the input data and through a `:wfRoleInt` synchronisation channel, and the result is eventually received from the executing conversation and stored in the final place. Note that Figure 7 only shows the workflow, and not the conversation. The meta relations between the metamodel and the depicted model are highlighted by means of dashed boxes and arrows. Moreover, only the most important relations are linked in the figure, while the others are not. However, it would be highly intuitive to link the remaining relations. Like all models, the DENEB's workflow model is composed of a number of distinct modeling elements, which are compliant with the reference nets terminology: *places*, *transitions*, *arcs* and their corresponding text *inscriptions*. These different elements, along with the way in which they are related, are defined in the scope of the workflow metamodel.

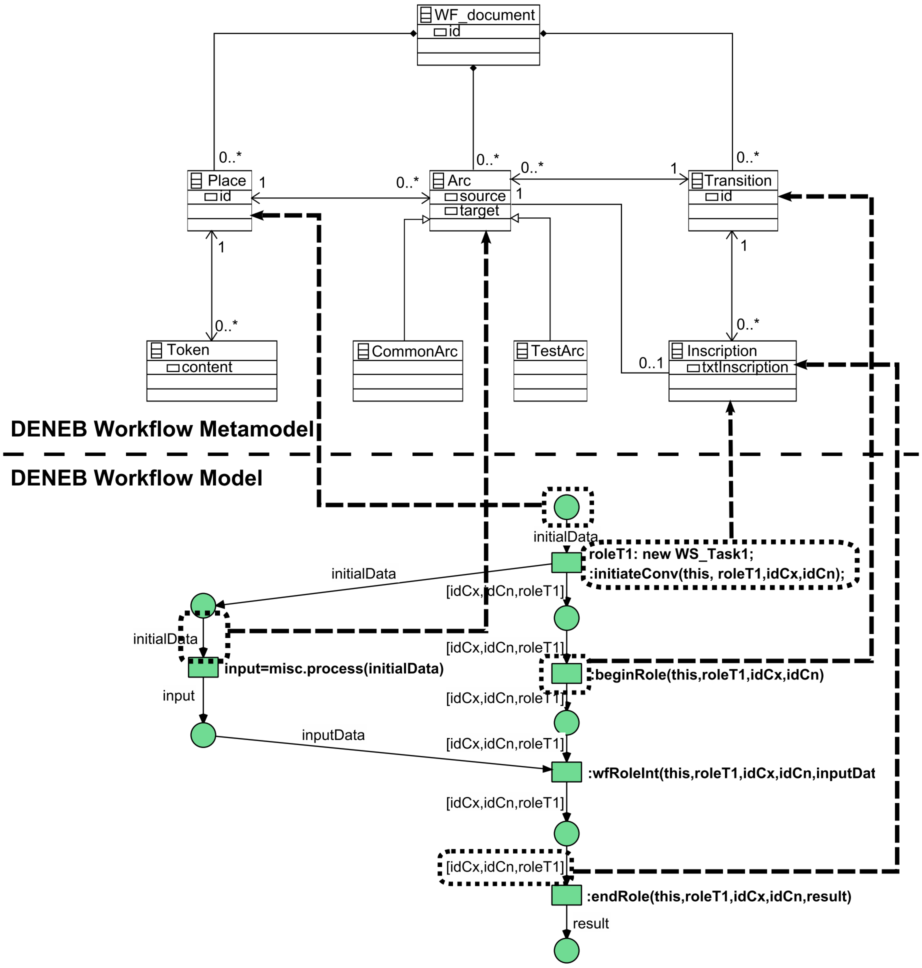


Figure 7: Conformance and meta relations with regard to the DENEBs workflow metamodel.

### *4.2. Model transformation and mapping implementation*

In order to develop the plug-in needed to carry out a specific model transformation, in our case from the composition service model to the workflow model, it is necessary to code the transformation rules in an ATL program. Each rule specifies a source pattern and a target pattern, both at metamodel level. Once the ATL transformation has been executed, the ATL engine attempts to establish a matching between the source pattern and the source model. Then, for each matching of the target pattern in the target model, the corresponding and specified code is generated for the target workflow model. As is shown in Figure 4, the ATL plug-in *ServiceComp2Workflow* is used to generate the corresponding workflow model, starting from any specific composition service model and conforming to the workflow metamodel. Target models will then be directly executed using the DENEB framework with no additional transformation, since Reference nets are directly executable on that approach.

The mappings have been defined by following the method shown below:

(i) Mappings between models are defined using natural language.
(ii) These mappings are structured by collecting them in a set of rules, again expressed in natural language.
(iii) Mapping rules are implemented using one of the existing model transformation approaches. In this case we have chosen the ATL Language.

Table 8 shows the mappings defined between the composition service model (source) and the workflow model (target) in a set of mapping rules defined in natural language. In order to clarify the description of the mappings, we have specified which elements in the source and target models are affected by each transformation, and have also provided some partial diagrams that will assist the reader to understand the rules.

Since the use of SOD-M executable nodes represents an outstanding difference with regard to other transformations approaches, we shall focus on the translation of these executable nodes. They normally represent service invocations and are used to access an external service provider in the composition service model, but can also represent internal invocations to an object method or a human intervention, depending on the value of the type tag. We shall focus on the first case for the sake of simplicity.

From the point of view of DENEB, the translation of an executable node generates the following two results. First, the role net required to execute the interaction with the invoked service. And second, a net skeleton is introduced in the workflow of the resulting DENEB process to initiate and manage the execution of the role net. On the other hand, from an execution point of view, both nets will be executed in a coordinated manner by means of the synchronised firing of a set of special transitions. A detailed explanation of these transitions will be provided in the following example.

We shall now explain a specific example of translation. In this case the executable node represents a REST interaction with an external service to validate a credit card. The left-hand side of Figure 9 shows the net skeleton introduced in the workflow while the right-hand side shows the role net required for the interaction (the *Role-REST_mediator* net). This role net is able to interact (by means of the message broker) with the REST mediator of DENEB which is responsible for invoking REST services. All the information required (endpoint, operation, parameters, etc.) has been specified in the executable node in the SOD-M composition service model.

A detailed explanation of the transitions of the resulting nets and the actions triggered by their execution is shown as follows. The role net is instantiated

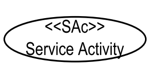
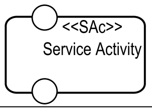
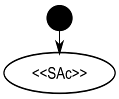
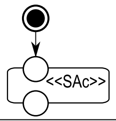
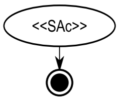
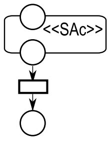
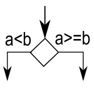
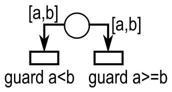
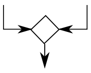
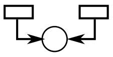
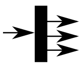
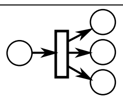
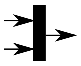
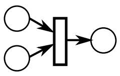
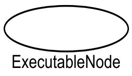
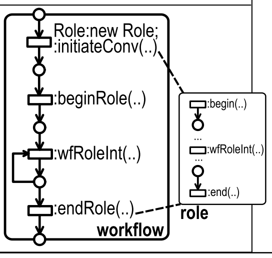
| Source Model | Source Elements | Description | Target Elements | Target Model |
|---|---|---|---|---|
| Service Composition Model |  | For each ServiceActivity in the source model, a DENEB's labeled block is generated. This block contains an input place and an output place. |  | DENEB Model |
| |  | For each ServiceActivity initial mark an initial place linked to the input place of the ServiceActivity target block is generated. This initial place is marked with a token. |  | |
| |  | For each ServiceActivity final mark an ending place linked from the output place of the ServiceActivity target block is generated. |  | |
| |  | For each decision node in the source model a conditional/guarded fork branch structure is generated. Guard inscriptions enable each possible branch transition according to source expressions. |  | |
| |  | For each decision join in the source model a join structure is generated, merging the different branches into a final place in the target model. |  | |
| |  | For each fork node in the source model a fork structure is generated. This structure contains as many parallel branches as paths are defined in the source model. |  | |
| |  | For each join node in the source model a join structure is generated in the target model, merging the branches correspondent to the source merging paths. |  | |
| |  | For each executable node in the source model a complete DENEB's protocol is generated, composed of a workflow skeleton and a role to be instantiated.<br>This structure is built using the tagged values (endpoint, operation, etc.) from the source node, and normally keeps a generic schema.<br>The workflow and the role interact by means of synchronization channels and through the main system net of DENEB. |  Role:new Role;<br>:initiateConv(..)<br>:beginRole(..)<br>:wfRoleInt(..)<br>:endRole(..)<br>workflow<br>:begin(..)<br>:wfRoleInt(..)<br>:end(..)<br>role | |

Figure 8: SOD-M to DENEB model transformation rules.

when the workflow fires a transition with the inscription `:initiateConv(..)`. In our example, this takes place by means of the transition `t02` in the workflow skeleton. The syncrhronised firing of transitions `:beginRole()` (`t03` and `t111` in the workflow and the role, respectively) then starts the role execution (in this case, a REST request to validate a credit card through the bank API). The necessary parameters (operation, URL, parameters) are passed from the workflow to the role by means of the `:wf-roleInt` synchronization channel inscribed in transitions `t04` and `t112`. These parameters will have the format of a tuple, such as `["REST", "validateCC","https://ws.bank.com/rest/", "skey=xxxx","ccid=22678856445656", "ccv=331", "cardholder=John Doe"]`. The REST interaction is then executed in the role by means of the interaction with the REST mediator that is integrated into the message broker (transitions `t113` and `t114`). Finally, the result from the REST execution is sent back to the workflow by means of the syncronised firing of transitions `:endRole()` (`t115` and `t05` in the role and the workflow nets, respectively). This firing also ends the execution of the role. Further information about how mediators work can be found in [13, 65]. It is supposed that the bank validation service will have established the format of the response message in its REST API, so no post-processing of the data sent back to the role is necessary.

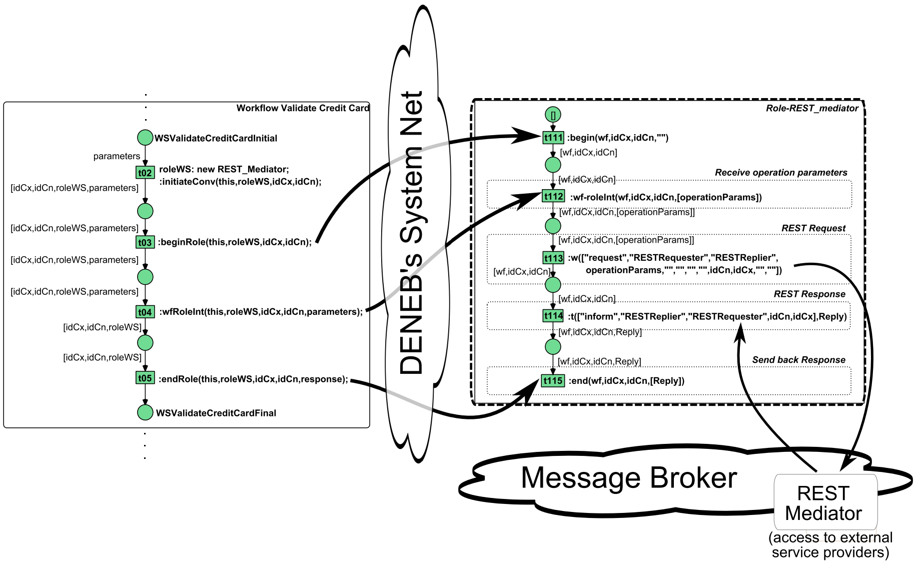

Workflow Validate Credit Card
WSValidateCreditCardInitial
parameters
t02
roleWS: new REST_Mediator;
:initiateConv(this,roleWS,idCx,idCn);
[idCx,idCn,roleWS,parameters]
[idCx,idCn,roleWS,parameters]
t03
:beginRole(this,roleWS,idCx,idCn);
[idCx,idCn,roleWS,parameters]
[idCx,idCn,roleWS,parameters]
t04
:wfRoleInt(this,roleWS,idCx,idCn,parameters);
[idCx,idCn,roleWS]
[idCx,idCn,roleWS]
t05
:endRole(this,roleWS,idCx,idCn,response);
WSValidateCreditCardFinal
DENEB's System Net
Role-REST_mediator
[]
t111
:begin(wf,idCx,idCn,"")
[wf,idCx,idCn]
Receive operation parameters
[wf,idCx,idCn]
t112
:wf-roleInt(wf,idCx,idCn,[operationParams])
[wf,idCx,idCn,[operationParams]]
REST Request
[wf,idCx,idCn,[operationParams]]
t113
:w(["request","RESTRequester","RESTReplier",
operationParams,"","","","",idCn,idCx,"",""])
[wf,idCx,idCn]
REST Response
[wf,idCx,idCn]
t114
:t(["inform","RESTReplier","RESTRequester",idCn,idCx],Reply)
[wf,idCx,idCn,Reply]
Send back Response
[wf,idCx,idCn,Reply]
t115
:end(wf,idCx,idCn,[Reply])
Message Broker
REST
Mediator
(access to external
service providers)

Figure 9: Workflow-role (REST mediator) interaction in DENEB.

Finally, it is also important to explain how the translation of guards from SOD-M to DENEB is carried out. Table 8 depicted an example in which the guards are translated from the SOD-M model to the DENEB's workflow. As was shown, DENEB requires all the variables involved to be bound to the input arcs of the guarded transition. In this approach we use a subset of XPath 1.0 to support expressions [66, 65].

Expressions are normally codified as a text string by means of natural lenguage. This string is then parsed and the equivalent XPath expression is generated. The subset considered here includes arithmethic operators (+, -,

*, div, mod), relational operators (=, !=, <=, <, >=, >) and logic operators (and, or). Precedency and associativity is also established among these operators. Equivalence and comparison string operators are also available (strcompare, strgt, strlt). The resulting XPath expression can be directly evaluated in DENEB by means of its built-in XPath engine.

It would be also possible to include functions and external calls in the guard expression. However, in such a case the corresponding library must be provided to load the required functions or procedures into the DENEB framework. Additional aspects regarding the dynamic load and extension of the DENEB framework can be viewed in [13].

## 5. Case Study

The transformation process depicted in this paper is shown in this section by means of its application to a real use case. First, the GMF modelling tool was used in order to build the initial SOD-M composition service model. The ATL transformation rules depicted in the previous section were then applied to generate the DENEB workflows. Eventually, these models (represented as DENEB nets) were imported and executed in the DENEB platform. The use case is presented first and the DENEB nets generated and the correspondence of their elements with the source composition service model are then described. Finally, some of the ATL rules involved in the transformation process are detailed.

### *5.1. Scenario*

The case study presented in this paper consists of a *Web IS for medical image management and processing* called GesIMED which has been developed in collaboration with the GTEBIM research group at Rey Juan Carlos University (RJCU) [67, 68].

Nowadays, medical image modalities such as Positron Emission Tomography (PET) and Functional Magnetic Resonance Imaging (FMRI) allow images of quantitative and qualitative cerebral activity to be obtained. Neuroscience researchers, particularly those working in clinical fields such as neuroradiology, neurology, neuropsychiatry and neuropsychology, carry out FMRI and PET studies. The first step towards achieving their work is to obtain the required images. They then process them using different techniques, and in very different manners related to improvement, correction and statistical analysis.

GesIMED is a Web IS which has been successfully deployed in several medical and research centres in Madrid, Spain, facilitating the work of neuroscientists and clinicians. Its objective is to provide neuroscience researchers with (a) *an easy-to-access historical medical image store* and (b) *a duly standardized method for processing and analysing the images.* It also stores both the original and the processed images in a database. Figure 10 shows a layout of the GesIMED system.

Neuroscience researchers are offered three specific services:

- **Storage Service.** The storage service is implemented by a database which stores the images and the results of the processes performed on them. Neuroscience researchers may access and consult them using different criteria (such as study, pathology, etc.) for diagnostic purposes, research or even teaching.

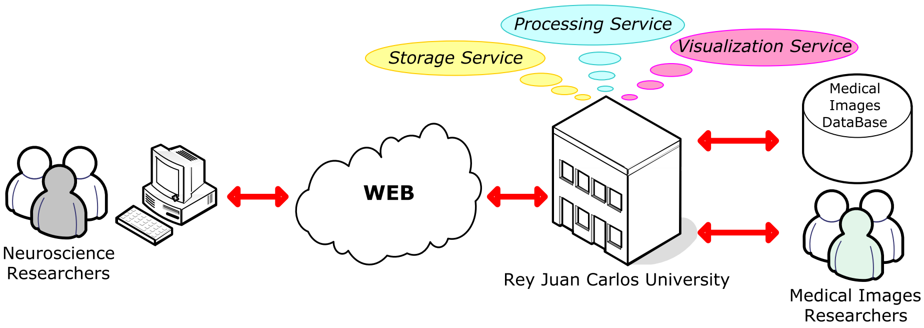


Figure 10: Layout of GesIMED system.

- **Processing Service.** Neuroscience researchers are offered two kinds of processing: analysis and segmentation. The former consists of the measurement of physical and physiological parameters, while the latter refers to image segmentation. A typical example of image segmentation would be the extraction of the bone's part from a radiological image. Researchers can request to process a set of images stored in the database or upload new medical images to be processed.

- **Visualization Service.** The visualization service provides image reconstruction processes (for example, the creation of 3D images from 2D images) and multimodality functions (such as the creation of both anatomical and functional image views within a single image). Researchers can follow the same procedures as in the processing service, to request visualizations of medical images stored in the database or to upload new images in order to view them later.

This case study has already been fully modelled using the SOD-M method in previous works [67, 12]. In this work we shall only present the composition service model, along with the resulting workflow model that was generated from the transformation process. This workflow model can be imported to the DENEB framework and executed without additional transformations. Both models for the case study are first described, and the implemented model transformation result is then presented.

### *5.2. Models*

Figure 11 shows the *composition service model* of the GesIMED case which represents the composition process of the different activities needed to carry out the *obtain processed images* business service. This model was built by taking the elements identified in a previous service process model, along with certain elements identified in the value model (such as business actors) as input.

As shown, the composition service model contains three different collaborators: the LAIM, the *processing service* (SPIm), and the *storage service* (SACim). The LAIM collaborator is decomposed into several service activities. First, a *login* is required to access the business service. Once the user's credentials have been verified, the *Pay Processing Fee* SAc starts a parallel processing: the payment is registered in the system (*Register Pay* action) and the information from

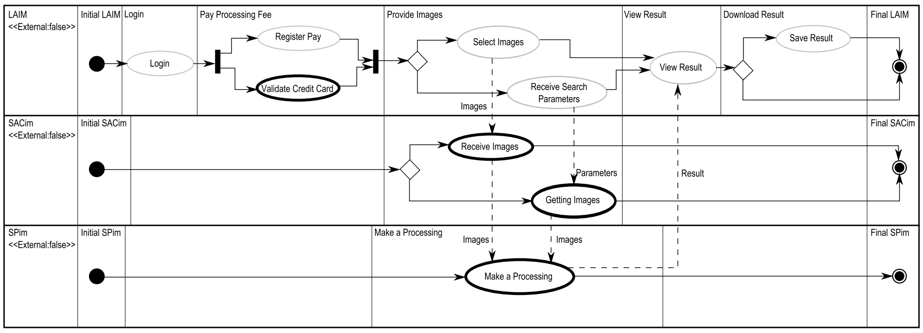


Figure 11: Composition service model for the business service *obtain processed images*.

the credit card is validated (*Validate Credit Card* action). When this has finished, there are two different possibilities depicted in the *Provide Images* SAc. On the one hand, the user can select a set of images in order to retrieve them (*Select Images*). On the other hand, the user can specify certain search parameters in order to obtain the selected images (*Receive Search Parameters*). These two actions exchange some information with the actions contained in the SACim collaborator (*Receive Images* and *Getting Images*, respectively), which will interact with the *Make a Processing* action in the SPim collaborator. As a result, the data generated will be obtained by means of the *View Result* action, and the user has the option of choosing to download and save this information or simply to exit from the process.

The *validate credit card* executable node represents a Web service since it is already a standard Web service provided by several organizations. Moreover, the activities performed by both the SPIm and the SACim are represented as Web services, since their behaviour is related to the communication between the central server of the application and the different services provided by them. As was previously described in Section 4, these types of nodes have a tagged value which contains the set of operations they perform and the information required to achieve the Web service invocation on a concrete platform (`endpoint`, `operation name`, `input` and `output` parameters).

An extract (corresponding to the *Initial LAIM*, *Login* and *Pay Processing Fee* SAcs) from the result of the transformation of the LAIM collaborator for the *obtain processed images* process is depicted in Figure 12. This is the resulting DENEB workflow as seen in the tool once it has been imported. A layout schema filter is applied in order to beautify the net, thus producing a printable net but without altering its execution capabilities. Figure 12 also shows the elements corresponding to the *register pay* activity operation and the *validate credit card* Web service elements in the composition service model. The credit card validation service is the most interesting, since it depicts the service-oriented capabilities of the approach presented in this work.

Let us now concentrate on the case shown in Figure 12. First, the *Pay Processing Fee* service activity generates a block in the workflow model labeled with the `<<SAc>> Pay Processing Fee` inscription. Then, both the *register pay* and the *validate credit card* elements contained in that activity generate the two corresponding blocks placed inside the `<<SAc>> Pay Processing Fee`. As is

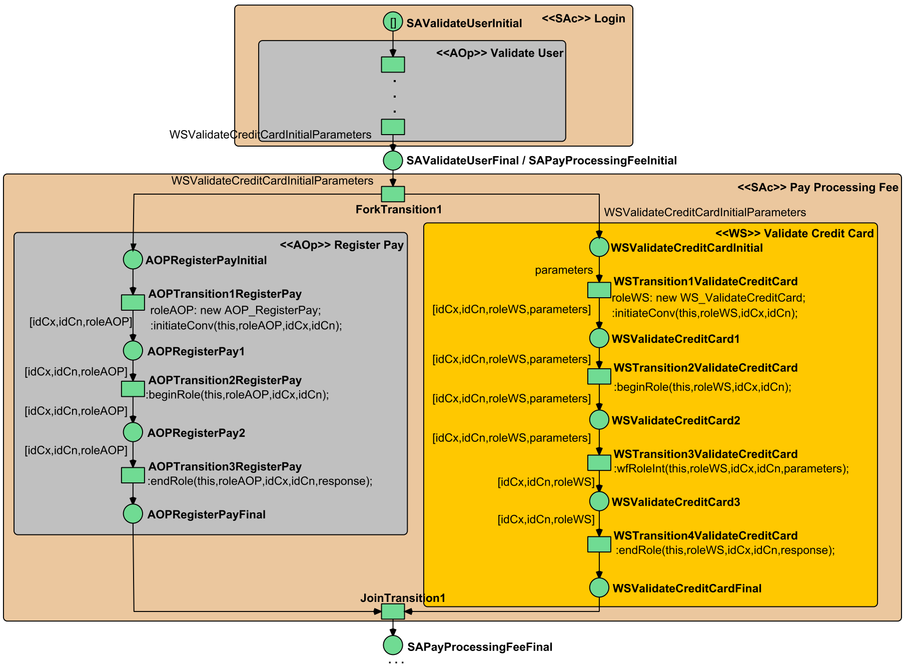

[]
SAValidateUserInitial
<<SAc>> Login
<<AOp>> Validate User
WSValidateCreditCardInitialParameters
SAValidateUserFinal / SAPayProcessingFeeInitial
WSValidateCreditCardInitialParameters
<<SAc>> Pay Processing Fee
ForkTransition1
WSValidateCreditCardInitialParameters
<<AOp>> Register Pay
AOPRegisterPayInitial
AOPTransition1RegisterPay
roleAOP: new AOP_RegisterPay;
:initiateConv(this,roleAOP,idCx,idCn);
[idCx,idCn,roleAOP]
AOPRegisterPay1
[idCx,idCn,roleAOP]
AOPTransition2RegisterPay
:beginRole(this,roleAOP,idCx,idCn);
[idCx,idCn,roleAOP]
AOPRegisterPay2
[idCx,idCn,roleAOP]
AOPTransition3RegisterPay
:endRole(this,roleAOP,idCx,idCn,response);
AOPRegisterPayFinal
<<WS>> Validate Credit Card
WSValidateCreditCardInitial
parameters
WSTransition1ValidateCreditCard
roleWS: new WS_ValidateCreditCard;
:initiateConv(this,roleWS,idCx,idCn);
[idCx,idCn,roleWS,parameters]
WSValidateCreditCard1
[idCx,idCn,roleWS,parameters]
WSTransition2ValidateCreditCard
:beginRole(this,roleWS,idCx,idCn);
[idCx,idCn,roleWS,parameters]
WSValidateCreditCard2
[idCx,idCn,roleWS,parameters]
WSTransition3ValidateCreditCard
:wfRoleInt(this,roleWS,idCx,idCn,parameters);
[idCx,idCn,roleWS]
WSValidateCreditCard3
[idCx,idCn,roleWS]
WSTransition4ValidateCreditCard
:endRole(this,roleWS,idCx,idCn,response);
WSValidateCreditCardFinal
JoinTransition1
SAPayProcessingFeeFinal
...

Figure 12: Workflow excerpt corresponding to the *obtain processed images* service, and generated as a result of the transformation process.

shown, DENEB workflows not only model the flow control of the execution, but must also follow a defined structure. These blocks contain a predefined structure which corresponds to a role instantiation and a workflow-role interaction in DENEB.

Once the workflow model has been generated, it can be imported to the DENEB platform and directly executed. The workflow depicted in Figure 12 will be placed in the workflow engine and will start its execution firing the enabled transitions at each moment. As shown, the initial token mark is placed in the *Login* service activity, which will be the first task to be executed. Once the login has been completed, the `ForkTransition1` transition will fire, and two parallel branches will then be under execution, corresponding to the *register pay* activity operation and the *validate credit card* Web service, respectivelly.

All those activities that take place by following the conversational approach (that is, the interaction logic), must be instantiated in the workflow by means of what are denominated as the *synchronisation channels* in DENEB's terminology. First, the corresponding role that must be played by the workflow is instantiated at runtime using the `:new <rolename>` inscription (*AOPTransition1RegisterPay* and *WSTransition1ValidateCreditCard* transitions). The new instance is then passed to the system net, which is the core of DENEB, through the synchronised firing of transitions labelled with the inscription `:initiateConv(..)`. The normal life cycle of the roles in DENEB is therefore the following:

(i) start the execution of the instantiated role (channel `:beginRole(..)`, in *AOPTransition2RegisterPay* and

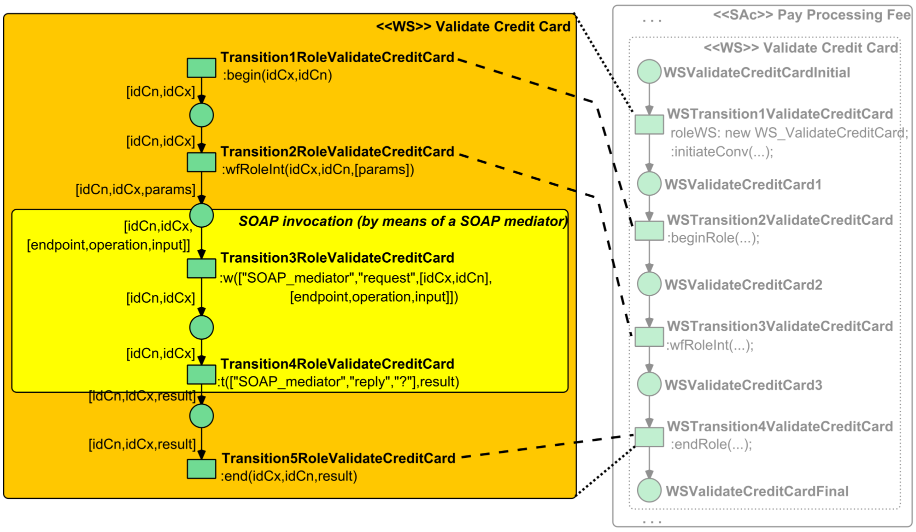
*WSTransition2ValidateCreditCard* transitions);

(ii) if required, workflows and roles can exchange information through the system net using the channel `:wfRoleInt(..)` as many times as required (for example, in the *WSTransition3ValidateCreditCard* transition, this mechanism is used to pass the parameters that are required to make the corresponding Web service invocation to the role);

(iii) finally, the roles end their execution and return the final result by means of the synchronised firing of transitions labelled `:endRole(..)` (*AOPTransition3RegisterPay* and *WSTransition4ValidateCreditCard* transitions).

A more detailed description of the DENEB system net and the described synchronisation channels which are in charge of managing the workflow-role interactions is provided in [57, 13].

Let us now concentrate on the explicit separation between activity operations and Web service invocations inside the *Pay processing fee* service activity depicted in the workflow. Whereas the *Register pay* operation corresponds to an internal operation, the *Validate credit card* operation requires a Web service invocation. A protocol playing the role side of requester or invocator of a Web service has therefore been automatically generated. As previously described, this protocol is put into execution by the workflow when the corresponding branch is reached. First, the firing of the *WSTransition1ValidateCreditCard* transition creates a new instance of the protocol and initiates it. Figure 13 depicts, on the left, the protocol implementation generated to interact with the *Validate credit card* Web service using SOAP and, on the right, the corresponding excerpt of the workflow involved in the interaction.



<<WS>> Validate Credit Card
Transition1RoleValidateCreditCard
:begin(idCx,idCn)
[idCn,idCx]
[idCn,idCx]
Transition2RoleValidateCreditCard
:wfRoleInt(idCx,idCn,[params])
[idCn,idCx,params]
SOAP invocation (by means of a SOAP mediator)
[idCn,idCx,
[endpoint,operation,input]]
Transition3RoleValidateCreditCard
:w(["SOAP_mediator","request",[idCx,idCn],
[endpoint,operation,input]])
[idCn,idCx]
[idCn,idCx]
Transition4RoleValidateCreditCard
:t(["SOAP_mediator","reply","?"],result)
[idCn,idCx,result]
[idCn,idCx,result]
Transition5RoleValidateCreditCard
:end(idCx,idCn,result)
. . .
<<SAc>> Pay Processing Fee
<<WS>> Validate Credit Card
WSValidateCreditCardInitial
WSTransition1ValidateCreditCard
roleWS: new WS_ValidateCreditCard;
:initiateConv(...);
WSValidateCreditCard1
WSTransition2ValidateCreditCard
:beginRole(...);
WSValidateCreditCard2
WSTransition3ValidateCreditCard
:wfRoleInt(...);
WSValidateCreditCard3
WSTransition4ValidateCreditCard
:endRole(...);
WSValidateCreditCardFinal
. . .

Figure 13: Corresponding protocol (on the left) for the *Validate credit card* Web service activity in the workflow (on the right).

The synchronised firing of the *WSTransition2ValidateCreditCard* transition (in the workflow) and the *Transition1RoleValidateCreditCard* transition (in the protocol) starts the protocol execution itself. The information related to the credit card data is passed from the workflow to the role by means of the `:wfRoleInt()` channel, thus firing the *WSTransition2ValidateCreditCard* and *Transition2RoleValidateCreditCard* transitions, respectively. Once this information has been passed, the workflow keeps the corresponding branch blocked until a response from the protocol is obtained. The protocol also performs a SOAP invocation through the message broker using the *write* and *take* directives in order to pass the required information to the SOAP mediator and to obtain the corresponding answer, firing the *Transition3RoleValidateCreditCard* and *Transition4RoleValidateCreditCard* transitions. The response is then passed to the workflow as an ending message by firing the *Transition5RoleValidateCreditCard* transition, and the protocol ends its execution. Note that the error control branches have been hidden in order to clarify the presentation of the resulting nets. The workflow then receives the response from the protocol and resumes its execution, joining both execution branches by firing the *JoinTransition1* transition, and continuing the execution of the business process.

The skeleton generated in the workflow model remains the same for both a SOAP or a REST case, and changes are only introduced in the corresponding role. A similar *Validate Credit Card* Web service activity using REST was depicted previously at the end of Section 4. As shown, neither the SOAP nor the REST roles contain any technological detail. This is owing to the use of the mediation system in the DENEB message broker. The SOAP and REST mediators allow this type of models to be produced, while avoiding such specific details, thus increasing flexibility and providing a very suitable mechanism to promote reusability and optimization.

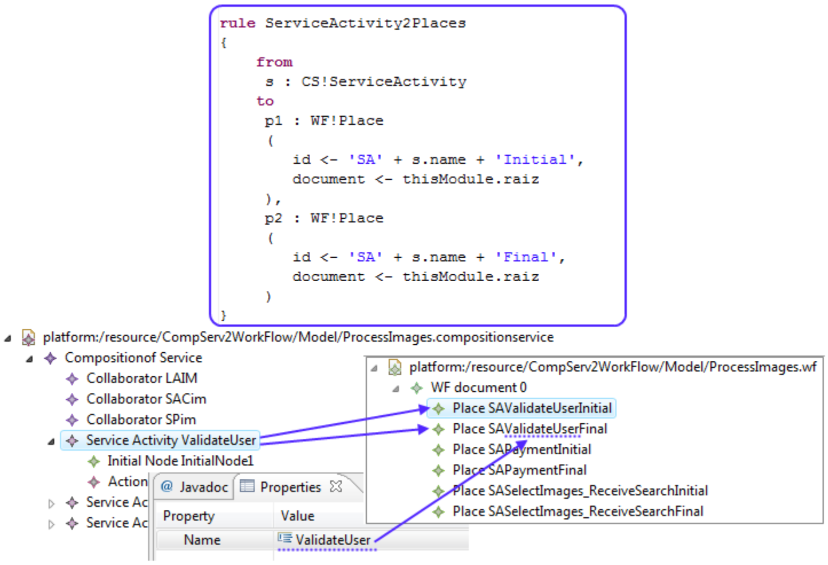
Let us now briefly depict the case of the *Register Pay* action in the workflow depicted in Figure 12. The code for this action must access an internal organizational resource, that is, a database, to register each payment which has been made. The workflow excerpt generated is almost equal to a service invocation, and also the corresponding role, which will interact with a database mediator. What is different in this case is the information exchanged between the workflow and the role. The corresponding mediator to access the database will require knowledge of the type of database, the query to be executed, the access policy information and the parameters of the query. DENEB provides a database mediator built on top of the Hibernate framework, thus supporting several database implementations using the same role. As has been shown, the capabilities of the DENEB framework allows us to provide a very flexible code for the implementation.

Finally, let us now describe some of the ATL transformation rules involved in the previous description. Figures 14 and 15 illustrate the model transformation and shows an excerpt from the *CompServ2Workflow* ATL transformation plus a partial view of *Process Images CompositionService* and *Workflow* models.

Figure 14 shows how the source pattern in the *ServiceActivity2Places* rule matches all the *ServiceActivity* object found in the source model (*CompositionService* model). The target pattern states that two *Place* objects are to be created in the target model (*Workflow* model) for each matching, one initial place and one final place in accordance with the transformation table depicted in Figure 8. The *Id* identifier for each new place in the target model is built using the chain `SA + <ServiceActivity name> + Initial or Final>`, thus generating unique names for these identifiers. The `document <- thismodule.raiz` instance is used to relate the `Place` element to the root element in the model

(`WF document`). `Raiz` represents the root element and it is a Helper, codified in ATL, that locates the root element in the target model, while `document` is the name of the relation between the `Place` element and the root.

For example, as is shown in Figure 14, the *ValidateUser* service activity contained in the *Login* AOP of the service composition model (Figure 11) corresponds to the excerpt contained in the *SAc Login* box in the generated workflow (top of Figure 12). As shown, two places have been generated: the initial place *SAValidateUserInitial* and the final place *SAValidateUserFinal* (which also corresponds to the *SAPayProcessingFeeInitial* initial SA place). The *ValidateUser* action is then contained between these two places in the target workflow model.



```
rule ServiceActivity2Places
{
    from
     s : CS!ServiceActivity
    to
     p1 : WF!Place
      (
         id <- 'SA' + s.name + 'Initial',
         document <- thisModule.raiz
     ),
     p2 : WF!Place
      (
         id <- 'SA' + s.name + 'Final',
         document <- thisModule.raiz
      )
}
```

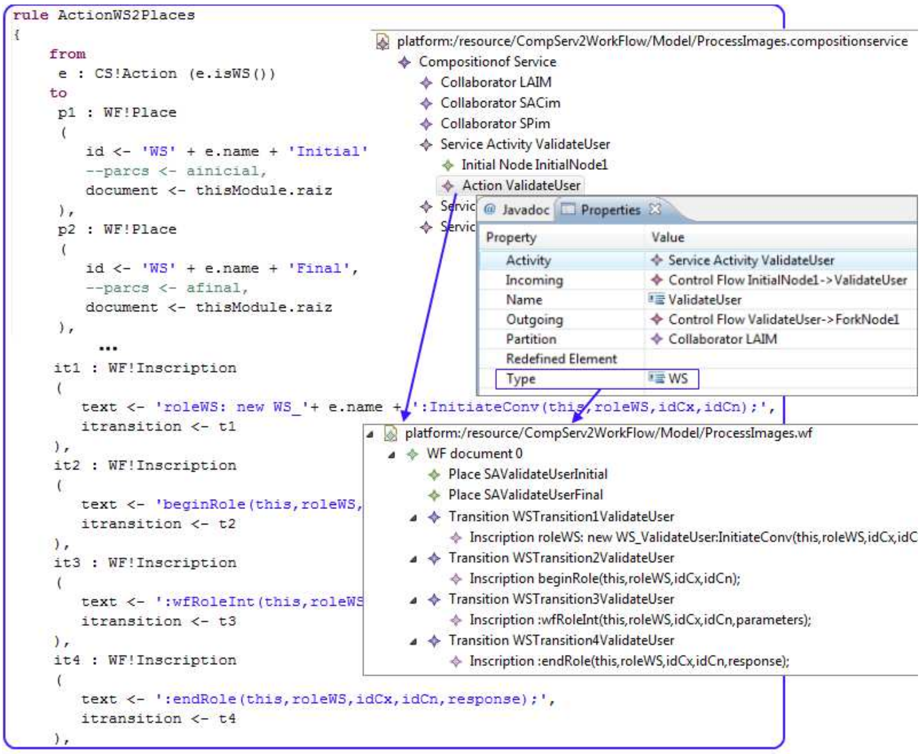
platform:/resource/CompServ2WorkFlow/Model/ProcessImages.compositionservice
- Compositionof Service
  - Collaborator LAIM
  - Collaborator SACim
  - Collaborator SPim
  - Service Activity ValidateUser
    - Initial Node InitialNode1
    - Action
  - Service Ac
  - Service Ac

| Property | Value |
|---|---|
| Name | ValidateUser |

platform:/resource/CompServ2WorkFlow/Model/ProcessImages.wf
- WF document 0
  - Place SAValidateUserInitial
  - Place SAValidateUserFinal
  - Place SAPaymentInitial
  - Place SAPaymentFinal
  - Place SASelectImages_ReceiveSearchInitial
  - Place SASelectImages_ReceiveSearchFinal

Figure 14: *CompService2Workflow* ATL code excerpt, rule *ServiceActivity2Places*.

On the other hand, Figure 15 partially shows the *ActionWS2Places* rule for which some of the objects (places, transitions and inscription) are created in the target model for every *Action* object of type WS found in the source model. This ATL rule was responsible for the generation of the *WS Validate Credit Card* skeleton in the workflow (Figure 12) from the *ValidateCreditCard* WS action in the service composition model (Figure 11). As shown in Figure 15, and in accordance with the transformation rules depicted in Table 8, in the specific case of a *WS* action, a skeleton is automatically generated in the target workflow model. This skeleton is then configured with all the information required to instantiate a WS conversation, initiate it and play the participant's corresponding role. All the information regarding the WS is contained in the source SOD-M model. The way it is organised to play the corresponding role is achieved in the rule. The conversation corresponds to a pattern which interacts with the mediation system in DENEB, and which will process the required Web service interactions (a detailed overview of this interaction process was previously shown in Section 4).



```
rule ActionWS2Places
{
    from
     e : CS!Action (e.isWS())
    to
     p1 : WF!Place
      (
        id <- 'WS' + e.name + 'Initial'
        --parcs <- ainicial,
        document <- thisModule.raiz
     ),
     p2 : WF!Place
      (
        id <- 'WS' + e.name + 'Final',
        --parcs <- afinal,
        document <- thisModule.raiz
     ),
       ...
    it1 : WF!Inscription
     (
       text <- 'roleWS: new WS_'+ e.name + ':InitiateConv(this,roleWS,idCx,idCn);',
       itransition <- t1
    ),
    it2 : WF!Inscription
     (
       text <- 'beginRole(this,roleWS,
       itransition <- t2
    ),
    it3 : WF!Inscription
     (
       text <- ':wfRoleInt(this,roleWS
       itransition <- t3
    ),
    it4 : WF!Inscription
     (
       text <- ':endRole(this,roleWS,idCx,idCn,response);',
       itransition <- t4
    ),
```

platform:/resource/CompServ2WorkFlow/Model/ProcessImages.compositionservice
- Compositionof Service
  - Collaborator LAIM
  - Collaborator SACim
  - Collaborator SPim
  - Service Activity ValidateUser
    - Initial Node InitialNode1
    - Action ValidateUser
  - Servic
  - Servic

| Property | Value |
|---|---|
| Activity | Service Activity ValidateUser |
| Incoming | Control Flow InitialNode1->ValidateUser |
| Name | ValidateUser |
| Outgoing | Control Flow ValidateUser->ForkNode1 |
| Partition | Collaborator LAIM |
| Redefined Element | |
| Type | WS |

platform:/resource/CompServ2WorkFlow/Model/ProcessImages.wf
- WF document 0
  - Place SAValidateUserInitial
  - Place SAValidateUserFinal
  - Transition WSTransition1ValidateUser
    - Inscription roleWS: new WS_ValidateUser:InitiateConv(this,roleWS,idCx,idC
  - Transition WSTransition2ValidateUser
    - Inscription beginRole(this,roleWS,idCx,idCn);
  - Transition WSTransition3ValidateUser
    - Inscription :wfRoleInt(this,roleWS,idCx,idCn,parameters);
  - Transition WSTransition4ValidateUser
    - Inscription :endRole(this,roleWS,idCx,idCn,response);

Figure 15: *CompService2Workflow* ATL code excerpt, rule *ActionWS2Places*.

As shown in this section, the application of the approach presented herein to the GesIMED real use case has resulted in the generation of the corresponding high-level Petri nets which model executable DENEB workflows and protocols automatically. This signifies that the SOD-M methodology was applied from a very early development stage in order to obtain the required models. The composition service model was used to apply the plug-ins developed in order to transform this model and generate the corresponding nets. These nets were then imported to the DENEB platform and consequently deployed and executed without any human intervention.

## 6. Conclusions and Future Work

In this paper, the Service-Oriented Development Method, SOD-M, and the platform for the Development and Execution of iNteroperable dynamic wEB processes, DENEB, have been integrated. The result is a model driven framework for the analysis, design, development and execution of business processes which covers BPM solutions from the very early stages of their development to their deployment and execution. The integration of both approaches is based on the automatic generation of code that is directly executable in DENEB. The contribution of our proposal is the ability to generate executable code without human intervention, as opposed to code skeletons or fragments. This solution frees business analysts and IT developers from the implementation, deployment and execution details of business processes.

Moreover, the resulting framework has been tested in a real case study, GesIMED, a Web IS for medical image management and processing. Business analysts have used the SOD-M methodology to specify their business goals and requirements. Both of them have been expressed in a service composition model, and a set of transformation rules has subsequently been applied in an executable DENEB process. Our solution permits the implementation (code programming) stage to be eliminated and eases the deployment of the resulting business process in the target execution environment. The cost of the development has therefore been reduced and deployment risks have been minimized.

On the other hand, as was mentioned in the introduction, the analysis of process models is another relevant challenge in the MDE field. In this respect, our approach has the advantage of using Petri nets to program executable processes. The mathematically sound foundation of Petri nets allows different analysis techniques to be integrated into our approach for the validation and verification of process models. Although these solutions are not within the scope of this paper, we shall briefly discuss them and refer to our previous experience in the field of process analysis.

The analysis of models consists of checking whether the process modelled actually behaves as expected. Different techniques can be used for this purpose, such as simulation, enumeration or structure based analysis techniques. Simulation has traditionally been used for the validation of processes. In our approach, once a SOD-M model has been translated to a DENEB executable process, it can be simulated using the DENEB execution environment and its behaviour can therefore be studied and analyzed. Moreover, the simulation capabilities of DENEB could be reused to analyse and improve the performance of the resulting process. Performance models for Petri nets could be integrated into our workflows and roles ([69], for instance) to evaluate performance estimations of the process. On the other hand, we are also able to verify some behavioural properties of DENEB processes, in other words, to guarantee that processes are free of certain logical errors. In [70] we proposed a method with which to verify whether a DENEB process can correctly interact with other processes so that the process terminates correctly. An implementation of this method could be directly integrated into the solution presented in this paper. Other behavioural properties could be also checked by reusing the traditional verification techniques of Petri nets [71, 72]. [73] provides an interesting overview of the techniques and tools developed for Petri nets to be exploited in the verification of business processes modelled using the WS-BPEL standard.

Finally, in the near future we intend to increase the capabilities of our MDA framework to support semantic annotations of the SOD-M composition models. A given model, its activities, data, control structures and even the whole model will be semantically described using domain ontologies. This solution will be very interesting from an analytical point of view. In [74] we proposed a new class of Petri nets, *RDF Unary Petri nets* (RDF-U-PN), to model semantic business processes, along with a methodology based on model checking techniques for the verification of their correctness and behavioural properties. Semantic SOD-

M processes could be modelled using this Petri-net based formalism and then analyzed using the proposed method.

## Acknowledgments

This research has been supported by a grant from the Spanish Ministry of Science and Innovation (projects TIN2010-17905, TIN2008-03582 and CONSOLIDER CSD2007-0022). We would like to acknowledge Ángel Cortés Maya and Juan Ondiviela Julve for their collaboration in the implementation of the tool supporting SOD-M and the extra features developed for the DENEB platform.

## References

[1] A. Watson, A Brief History of MDA, Upgrade, The European Journal for the Informatics Profesional IX (2) (2008) 7–11.

[2] M. Havey, Essential Business Process Modeling, O'Reilly Media, Inc., 2005.

[3] OMG BPMN 2.0 - OMG Formal Specification (2011), Available at: http://www.omg.org/spec/BPMN/2.0/.

[4] BPEL 2.0 specification - OASIS, Available at http://docs.oasis-open.org/wsbpel/.

[5] L. Verner, BPM: The Promise and the Challenge, Queue 2 (1) (2004) 82–91.

[6] B. Selic, The pragmatics of model-driven development, IEEE Software 20 (5) (2003) 19–25.

[7] D. C. Schmidt, Model-driven engineering, IEEE Computer 2006, 39 (2).

[8] D. Ameller. Considering Non-Functional Requirements in Model-Driven Engineering, Master Thesis, Universitat Politecnica de Catalunya, Departament de Llenguatges i Sistemes Informatics, 2009.

[9] Miller, J., Mukerji, J., MDA Guide Version 1.0.1, Document number omg/2003-06-01. Available at: http://www.omg.com/mda, 2003.

[10] S. J. Mellor, A. N. Clark, T. Futagami, Model-Driven Development, IEEE Software 20 (2003) 14–18.

[11] B. Selic, MDA Manifestations, Upgrade, The European Journal for the Informatics Profesional IX (2) (2008) 12–16.

[12] V. De Castro, E. Marcos, R. Wieringa, Towards a service-oriented MDA-based approach to the alignment of business processes with IT systems: From the business model to a web service composition model, International Journal of Cooperative Information Systems 18 (2) (2009) 225–260.

[13] J. Fabra, P. Álvarez, J. Bañares, J. Ezpeleta, Runtime Protocol Binding: Flexible Service Integration By Means Of Flexible Service Interactions, in: 2008 International Conference on Services Computing – SCC 2008, LNCS, IEEE Computer Society Press, 2008, pp. 291–298.

[14] M. J. Ibáñez, P. Álvarez, J. Ezpeleta, Flow and Data Compatibility for the Correct Interaction among Web Processes, in: International Conference on Intelligent Agents, Web Technologies and Internet Commerce – IAWTIC'08, 2008, pp. 716–722.

[15] M.J. Ibáñez, P. Álvarez, J. Ezpeleta, Checking Necessary Conditions for Control and Data Flow Compatibility between Business and Interaction Logics in Web Processes, in: 6th IEEE European Conference on Web Services, 2008, pp. 92–101.

[16] Eclipse, Available at: http://www.eclipse.org/.

[17] F. Jouault, I. Kurtev, Transforming Models with ATL, in: MoDELS Satellite Events, 2005, pp. 128–138.

[18] E. Biermann, C. Ermel, G. Taentzer, Precise Semantics of EMF Model Transformations by Graph Transformation, 2008, pp. 53–67.

[19] J. Brunswick, Extending the Business Value of SOA through Business Process Management, Architect: SOA and BPM, 2008.

[20] C. Frye, BPM inside the belly of the SOA whale, Web Services News (2006) 1–3.

[21] F. Kamoun, A roadmap towards the convergence of business process management and service oriented architecture, Ubiquity 2007 (April) 1–1.

[22] Object Management Group (2010), Available at: http://www.omg.org/.

[23] V. De Castro, E. Marcos, M.L. Sanz, A model driven method for service composition modelling: a case study, International Journal on Web Engineering Technology 2 (4) (2006) 335–353.

[24] OMG Service Oriented Architecture SIG – ABSIG (December 2009), Available at: http://soa.omg.org/.

[25] R. K. Ko, S. S. Lee, E. W. Lee, Business Process Management (BPM) standards: A survey, Business Process Management journal, 2009 15 (5).

[26] S. Roser, B. Bauer, A Categorization of Collaborative Business Process Modeling Techniques, in: Seventh IEEE International Conference on E-Commerce Technology Workshops – CECW'05, IEEE Computer Society, 2005, pp. 43–54.

[27] OMG BPMN 1.1 - OMG Final Adopted Specification (2008), Available at: http://www.omg.org/docs/formal/08-01-17.pdf.

[28] P. Harmon, The OMG's Model Driven Architecture and BPM, Business Process Trends (2004) 1–3.

[29] ARIS Design Platform: Getting Started with BPM, 1st Edition, Springer, 2007.

[30] R. Johnson, Expert One-on-One J2EE Design & Development, Wrox Press Ltd., Birmingham, UK, 2002.

[31] A. Arkin, Business Process Modeling Language (BPML), Tech. rep., Business Process Management Initiative (BPMI) (2002).

[32] A. E. Walsh, Ebxml: The Technical Reports, Prentice Hall PTR, 2002.

[33] S. Thatte, XLANG: Web Services for Business Process Design, Tech. rep., Microsoft, Redmond, Wash. (2001).

[34] F. Leymann, Web Services Flow Language (WSFL 1.0), Tech. rep., IBM (May 2001).

[35] K. Sandy. BPM Think Tank. EbizQ, 2006, Available at: http://www.ebizq.net/blogs/column2/archives/bpmthinktank2006/.

[36] L. Dikmans, Transforming BPMN into BPEL: Why and How, Oracle Technology network.

[37] Bruce Silvers BPMS Watch, 2011, Available at http://www.brsilver.com/.

[38] S. A. White, Using BPMN to Model a BPEL Process. IBM Corp., United States.

[39] P. V. G. Remco Dijkman, BPMN 2.0 Execution Semantics Formalized as Graph Rewrite Rules, in: BPMN 2010, 2010.

[40] W.M.P. van der Aalst, A.H.M. ter Hofstede, YAWL: Yet Another Workflow Language, Information Systems 30 (4) (2005) 245–275.

[41] T. Murata, Petri nets: Properties, analysis and applications, in: Proceedings of IEEE, Vol. 77, 1989, pp. 541–580.

[42] P. Hrastnik, Execution of Business Processes Based on Web Services, International Journal of Electronic Business 2 (5) (2004) 550–556.

[43] N. Palmer, Understanding the BPMN-XPDL-BPEL Value Chain, Business Integration Journal (November/December 2006) (2006) 54–55.

[44] M. Brambilla, Generation of WebML web application models from business process specifications, in: 6th international conference on Web engineering – ICWE'06, ACM, 2006, pp. 85–86.

[45] Butler Group, Application Development Strategies. 2003.

[46] Unified Modeling Language (UML), Available at: http://www.uml.org/.

[47] S. J. Mellor, M. Balcer, Executable UML: A Foundation for Model-Driven Architectures, Addison-Wesley Longman Publishing Co., 2002.

[48] N. Guelfi, A. Mammar, A formal framework to generate XPDL specifications from UML activity diagrams, in: 2006 ACM symposium on Applied computing – SAC'06, ACM, 2006, pp. 1224–1231.

[49] P. Jiang, Q. Mair, J. Newman, Using UML to Design Distributed Collaborative Workflows: from UML to XPDL, IEEE International Workshops on Enabling Technologies (2003) 71.

[50] W. M. P. van der Aalst, Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management, in: Lectures on Concurrency and Petri Nets, Vol. 3098 of LNCS, Springer-Verlag, 2004, pp. 1–65.

[51] OMG and MDA: The Architecture of Choice for a Changing World, Available at: http://www.omg.org/mda/products_success.htm.

[52] V. De Castro, E. Marcos, J.M. Vara, Applying cim-to-pim model transformations for the service-oriented development of information systems, Information and Software Technology 53 (19) (2011) 87–105.

[53] E. Marcos, C. J. Acuña, C. E. Cuesta, Integrating software architecture into a mda framework, in: V. Gruhn, F. Oquendo (Eds.), 3rd European Workshop on Software Architecture – EWSA, Vol. 4344 of Lecture Notes in Computer Science, Springer, 2006, pp. 127–143.

[54] J. M. Vara, B. Vela, V. A. Bollati, E. Marcos, Supporting Model-Driven Development of Object-Relational Database Schemas: A Case Study, in: 2nd International Conference on Theory and Practice of Model Transformations – ICMT'09, Springer-Verlag, 2009, pp. 181–196.

[55] P. Cáceres, V. de Castro, J. M. Vara, E. Marcos, Model transformations for hypertext modeling on web information systems, in: 2006 ACM symposium on Applied computing – SAC'06, ACM, 2006, pp. 1232–1239.

[56] S. White, 2004, Process Modelling Notations and Workflow Patterns, IBM Corporation, available at: http://www.bpmn.org/Documents/Notations_and_Workflow_Patterns.pdf.

[57] J. Fabra, P. Álvarez, J. Bañares, J. Ezpeleta, A framework for the development and execution of horizontal protocols in open BPM systems, in: 4th International Conference on Business Process Management – BPM 2006, LNCS, Springer Verlag, 2006, pp. 209–224.

[58] J. E. Hanson, P. Nandi, S. Kumaran, Conversation support for business process integration, in: Sixth International Enterprise Distributed Object computing Conference – EDOC'02, IEEE Computer Society, 2002, p. 65.

[59] O. Kummer, Introduction to Petri nets and reference nets, Sozionik Aktuell 1 (2001) 1–9.

[60] J. Billington, S. Christensen, K. M. van Hee, E. Kindler, O. Kummer, L. Petrucci, The Petri Net Markup Language: Concepts, Technology, and Tools, in: International Conference on Applications and Theory of Petri Nets – ATPN 2003, Vol. 2679 of LNCS, Springer, 2003, pp. 483–505.

[61] O. Kummer, F. Wienberg, M. Duvigneau, M. Köhler, D. Moldt, H. Rölke, Renew – the Reference Net Workshop, in: 24th International Conference on Application and Theory of Petri Nets – ATPN 2003, 2003, pp. 99–102.

[62] J. Fabra, P. Álvarez, J. Ezpeleta, DRLinda: A Distributed Message Broker For Collaborative Interactions Among Business Processes, in: 8th International Conference on Electronic Commerce and Web Technologies – EC-Web'07, Vol. 4655 of LNCS, Springer Verlag, 2007, pp. 212–221.

[63] N. Carriero, D. Gelernter, Linda in context, Communications of the ACM 32 (4) (1989) 444–458.

[64] F. Budinsky, S. A. Brodsky, E. Merks, Eclipse Modeling Framework, Pearson Education, 2003.

[65] J. Fabra, P. Álvarez, BPEL2DENEB: Translation of BPEL Processes to Executable High-level Petri Nets, in: Fifth International Conference on Internet and Web Applications and Services – ICIW 2010, IEEE Computer Society Press, 2010.

[66] XML Path Language (XPath). Version 1.0 (1999), Available at: http://www.w3.org/TR/xpath/.

[67] J. A. H. Tamames, C. J. Acuña, V. de Castro, E. Marcos, M. L. Sanz, N. Malpica, Web-PACS for Multicenter Clinical Trials, IEEE Transactions on Information Technology in Biomedicine 11 (1) (2007) 87–93.

[68] GesIMED System (2008), Available at http://ariadna.escet.urjc.es/gesimed/.

[69] J. Merseguer, J. Campos, Software performance modelling using uml and petri nets, Lecture Notes in Computer Science 2965 (2004) 265–289.

[70] M. Ibáñez, P. Álvarez, J. Bañares, J. Ezpeleta, Control and Data Flow Compatibility in the Interaction between Dynamic Business Processes, Concurrency and Computation: Practice and Experience 23 (1) (2011) 57–85.

[71] W. Reisig, D. Fahland, N. Lohmann, P. Massuthe, C. Stahl, D. Weinberg, K. Wolf, K. Kaschner, Analysis techniques for service models, in: Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation, 2006 (ISoLA 2006), 15-19 November 2006, Paphos, Cyprus, IEEE Computer Society, 2006, pp. 11–17.
URL http://dx.doi.org/10.1109/ISoLA.2006.58

[72] W. M. P. van der Aalst, Workflow verification: Finding control-flow errors using petri-net-based techniques, in: Business Process Management, Models, Techniques, and Empirical Studies, Springer-Verlag, London, UK, 2000, pp. 161–183.
URL http://portal.acm.org/citation.cfm?id=647778.734905

[73] F. van Breugel, M. Koshkina. Models and Verification of BPEL. (2006), Draft published by the Department of the Computer Science and Engineering of the York University, available at http://www.cse.yorku.ca/ franck/research/drafts/tutorial.pdf.

[74] M.J. Ibáñez, P. Álvarez, J.A. Bañares, J. Ezpeleta. Analyzing Behavioral Properties of Semantic Business Processes with Parametric Data, Concurrency and computation: Practice and experience, accepted for publication, 2011.