Model Transformation for Service-Oriented Web Applications Development

Valeria de Castro, Juan Manuel Vara, Esperanza Marcos

Kybele Research Group
Rey Juan Carlos University
Tulipán S/N, 28933, Móstoles, Madrid, Spain
{valeria.decastro,juanmanuel.vara, esperanza.marcos}@urjc.es

Abstract. In recent years, innovation in technologies such as web services, business process automation, etc., have motivated a new paradigm in the application development field to appear, known as Service-Oriented Computing. This new paradigm, which utilizes services as fundamental elements for developing applications, has encouraged the evolution of web applications and the way they are developed. Attending to this evolution we have already presented a *model driven method for service-oriented web applications development*. The method defines new Platform Independent Models (PIMs) and mappings between them. The PIMs proposed have been grouped in a UML profile based on the behavioral modeling elements of UML 2.0. In this work, we focus on the mapping between those PIMs and we define the model to model transformations needed for service-oriented web applications development. We first specify the transformation rules with natural language to later formalize them with graph transformation rules.

Keywords. Service-Oriented Web Applications, MDA, UML, Model Transformations, Graph Transformation Rules.

1 Introduction

A new paradigm in the field of application development, known as *Service-Oriented Computing* (SOC) [12] has encouraged the evolution of web applications and the way they are developed. Thus, while first web applications were created as a way to make available information to users, and they were built basically by linking static and dynamic pages; currently, most of the web applications are understood as networks of applications owned and managed by many business partners providing several services satisfying the needs of consumers that pay for them. Services usually range from quite simple ones, like buying a book or renting a car to the ones which involve complex processes such as obtaining sales ratings or participating in a public auction. For that reason, in the Web Engineering field, there is a need for methodologies for development based on current technologies such as web services, business process execution, etc.

Although the design and implementation of web services can be apparently easy, the implementation of business processes using web services is not so effortless. Languages for the implementation of business processes have many limitations when they are used in the early stages of the development process [19]. This occurs mainly because the transformation from high-level business models generally carried out by business analysts; to a composition language that implements those business processes with web services is not a trivial issue.

Model Driven Architecture (MDA) [11] provides a conceptual structure where the diagrams used by business managers and analysts, as well as the various diagrams used by software developers can be fit. Moreover MDA allows organizing them in such a way that the requirements specified in one diagram can be traced through the more detailed diagrams derived from the former. Hence, MDA is a useful tool to anyone interested in aligning business processes with IT systems [8].

This paper deals with the MDA approach for the development of service-oriented web applications[1]. In a previous work we proposed a model-driven method which starts from a high level business model and allows obtaining a service composition model that makes easy the mapping to a specific web service technology [5]. To obtain this service composition model, which is represented through a UML activity model, the method defines: a Computational Independent Model (CIM) for business modeling, called value model [7]; four Platform Independent Models (PIMs) for the behavioral modeling of service-oriented web application; and mappings rules between them.

In this work we present *the metamodels of the PIMs defined by the method, which includes new elements for service-oriented web applications modeling* that extend the behavioral modeling elements of UML 2.0 [10]; and we focus on the *mapping rules between these PIMs*, which allows obtaining a service composition model that makes easy the mapping to a specific web service technology, starting form a high level UML use cases model in which the services required by the web consumers are represented.

Given that the method is based on a continuous development process in which, according to the MDA principles [9], the models act as the prime actors, mappings between models play a very important role. Each step of this process consists basically in the generation of an output model starting from one or more input models on which the mapping rules are applied. In this work, we follow a graph transformation approach to effectively realize the mappings between the PIMs proposed by the method. The term *Graph Transformation* is used to refer to a special kind of rule-based transformations that are typically represented diagrammatically [14]. So, given that the mappings were defined in a rule-based manner, it seems appropriate to use a graph transformation approach to later formalize them. A similar approach for object-relational database development was presented in a previous work [18].

The rest of the paper is structured as follows: section 2 presents the UML profile that includes the new elements for service-oriented web applications modeling at PIM level; section 3 describes the model to model transformations between the proposed

[1] This research is partially granted by the GOLD projects financed by the Ministry of Science and Technology of Spain (Ref. TIN2005-00010).

PIMs; finally, section 4 concludes the paper by underlying the main contributions and the future works.

2 UML profile for service-oriented web applications modeling

As mentioned before, the method proposed for service-oriented web applications development defines four new PIMs for modeling the behavioral aspect of web applications: the ***Business Services model***, the ***Extended Use Cases model***, the ***Services Delivery Process model*** and the ***Services Composition model***. Each one is defined through a metamodel that extends the UML metamodel [10]. Figure 1 shows the dependences of the new models proposed (shadowed in the figure) with respect to the UML packages for behavioral modeling. As shown in the figure, the models proposed in our method are represented through UML behavioral diagrams: while the business services model and the extended use cases model are represented through use cases diagrams, the services delivery process model and services composition model are represented through activity diagrams.

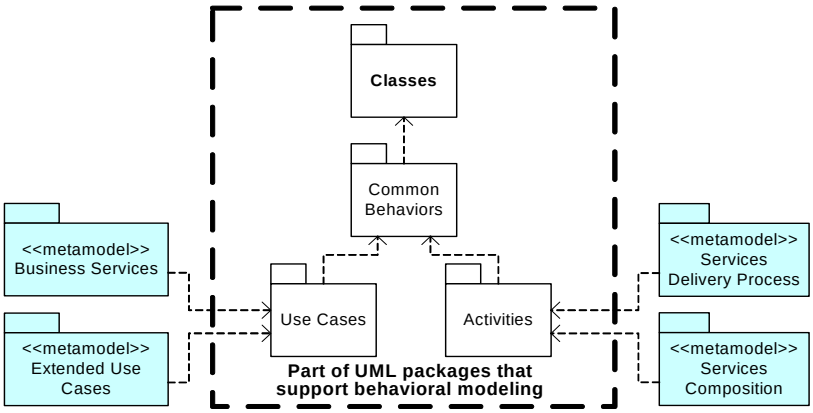


Fig. 1. Dependencies of new models regarding the UML packages for behavioral modeling

These new PIMs defined by the method include new modeling elements which have been grouped in a UML profile called MIDAS/BM (*MIDAS Behavior Modeling*). According to UML 2.0, a UML profile is a package that contains modeling elements that have been customized for a specific purpose or domain, using extension mechanisms, such as stereotypes, tagged definitions and constraints [10]. Our profile is defined over the behavioral modeling elements of UML 2.0 and it describes new elements for modeling the behavioral aspect of service-oriented web applications. Figure 2 shows the profile, including the newly proposed stereotypes that are applied over the existing metaclasses of the UML metamodel. The new stereotypes defined are described in Appendix A at the end of this document.

Next, we are going to present the metamodel of the new PIMs in which these elements are represented, to later describe the mapping rules between them. For the sake of space, we explain the metamodels by describing only the new elements defined,

the associations between them and the specification of the respective restrictions over these metamodels defined using the OCL standard. A complete example of how these models should be used can be found in [5].

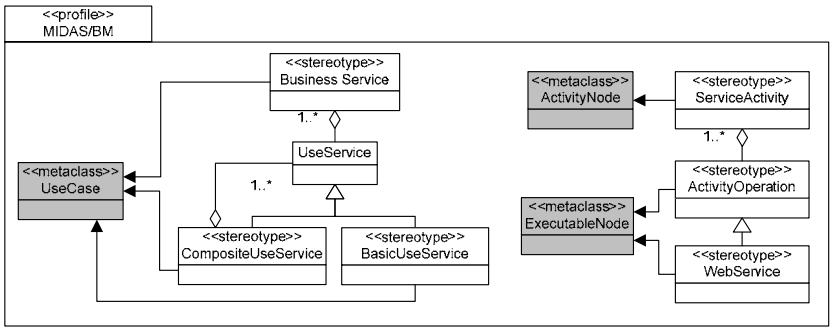


Fig. 2. The MIDAS/BM profile

Business Services Metamodel. The business service model is an extension to the UML use cases model in which only the actors and the *business services* that the system will provide them are represented. We define a business service as a complex functionality offered by the system, which satisfies a specific need of the consumer. The consumers of the system are represented in this model as actors. The business services are represented in this model as use cases stereotyped with <<*BusService*>> (see stereotype BusinessService in Appendix A).

Figure 3 shows the business services metamodel in which the new modeling element is shadowed. In the business services model each business service is associated to the actor needing the business service.

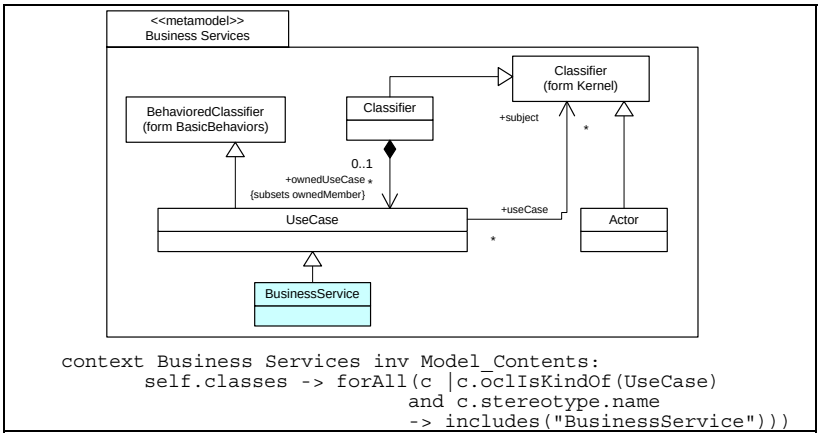
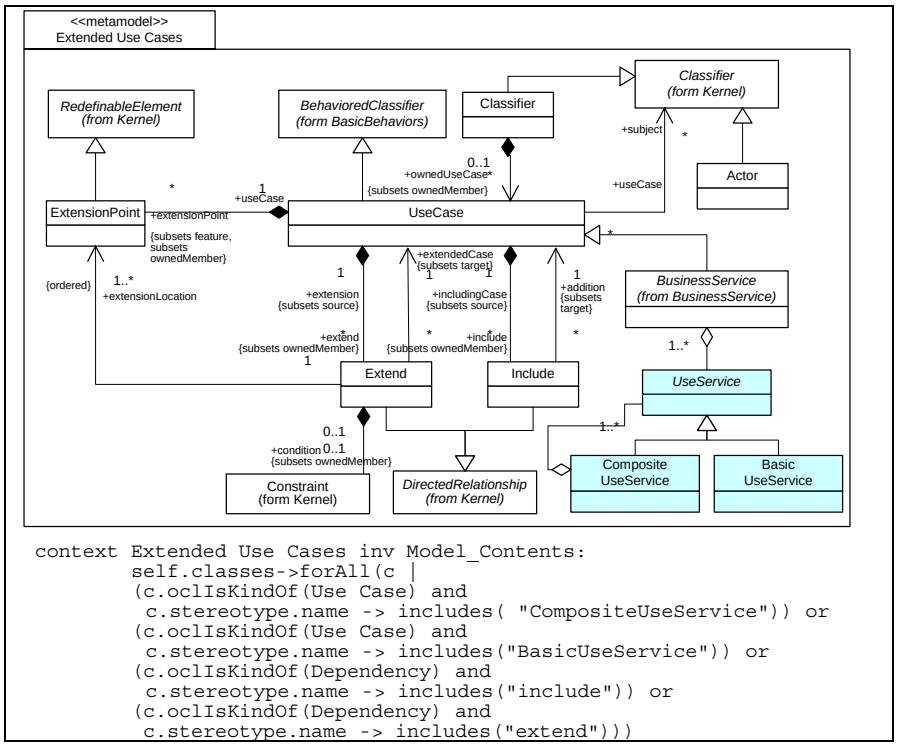


Fig. 3. Business services metamodel

Extended Use Cases Metamodel. This metamodel also extends the elements of the UML package for use cases modeling. In the extended use cases model we propose to represent the basic or composite *use services*. We define a use service as a functionality required by the system to carry out a business service. Thus, it represents a portion of the functionality of a business service. A *basic use service* is a basic unit of behavior of the web application, for instance 'registering as a costumer'. A *composite use service* is an aggregation of either basic or composite use services. The composite and basic use services are represented in this model as a special kind of UseCase stereotyped with <<*CS*>> and <<*BS*>> (see stereotypes CompositeUseService and BasicUseService in Appendix A).

Figure 4 shows the extended use cases metamodel in which the new modeling elements are shadowed. Note that UseService is an abstract class therefore it is not represented in the extended use cases model.




```
context Extended Use Cases inv Model_Contents:
        self.classes->forAll(c |
        (c.oclIsKindOf(Use Case) and
         c.stereotype.name -> includes( "CompositeUseService")) or
        (c.oclIsKindOf(Use Case) and
         c.stereotype.name -> includes("BasicUseService")) or
        (c.oclIsKindOf(Dependency) and
         c.stereotype.name -> includes("include")) or
        (c.oclIsKindOf(Dependency) and
         c.stereotype.name -> includes("extend")))
```

Fig. 4. Extended use cases metamodel

Services Delivery Process Metamodel. This metamodel extends the elements of the UML activity package. In the service delivery process model we propose to represent the activities that must be carried out for delivering a business service. The activities of this model are called *service activities.* The service activities are obtained transforming the basic use services identified in the previous model into activities of a

process. So, the services activities represent a behavior that is part of the execution flow of a business service. A service activity is represented as an ActivityNode stereotyped with <<*SAc*>> (see stereotype ServiceActivity in Appendix A).

The ServiceActivity element is shadowed in Figure 5 which shows the services delivery process metamodel.

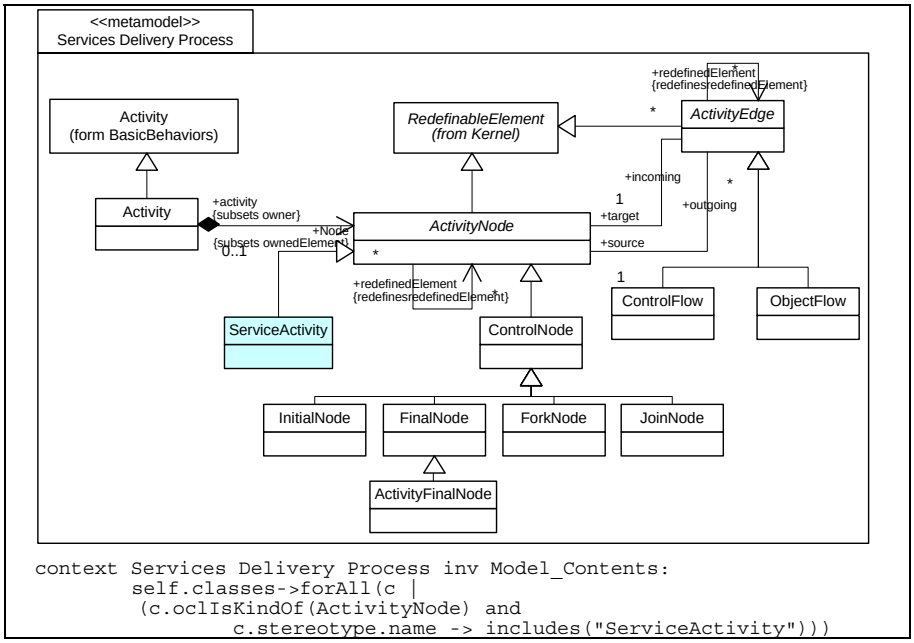


Fig. 5. Service delivery process metamodel

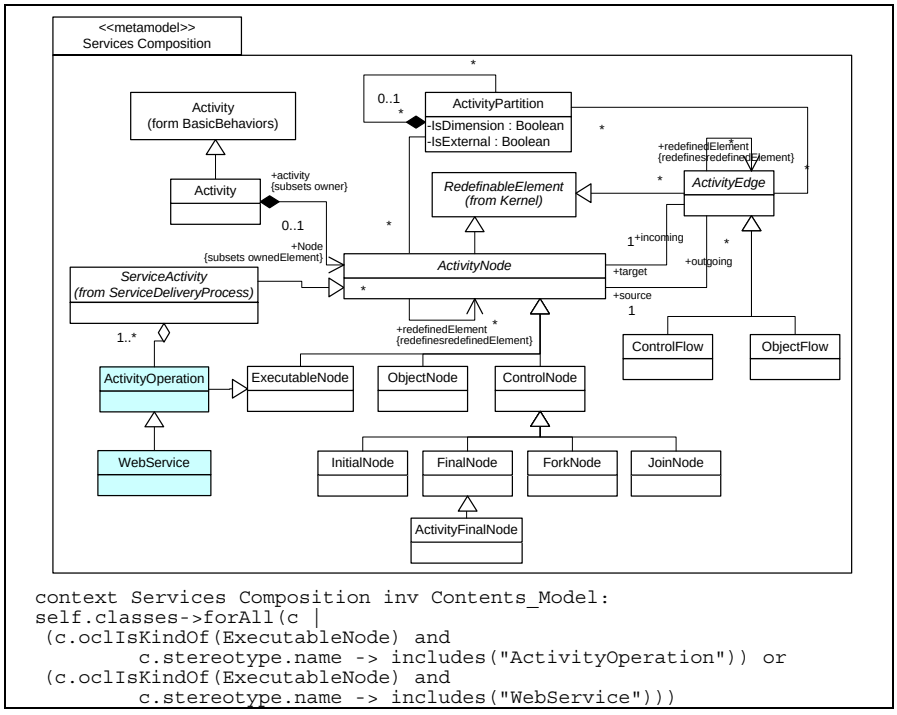
Services Composition Metamodel. This metamodel also extends the elements of the UML activity package. In this model we represent the execution flow of a business service too, but in a more detailed way by including the concepts: *activity operation* and *business collaborator*.

We define an *activity operation* as an action that is supported by the service activity. It is represented in this model as a special kind of ExecutableNodes stereotyped with <<*AOp*>> (see ActivityOperation in Appendix A). Additionally, the service composition model proposes to identify those activity operations that can be implemented as Web services, using a special kind of ExecutableNode stereotyped with <<*WS*>> (see stereotype WebService in Appendix A).

A *business collaborator* is defined as an organizational unit that carries out some activity operation which is involved in the services offered by the web application (i.e.: as a Web service). The business collaborators are represented in this model as ActivityPartitions, which can be indicated as a swim-lane in the activity diagram. The ActivityOperations and WebServices are distributed in ActivityPartitions according to the business collaborator that carries out the operation. A business collaborator can

be external to the system, in which case the ActivityPartition is labelled with the keyword «external».

Figure 6 shows the service composition metamodel, in which the new modeling elements are shadowed.




```
context Services Composition inv Contents_Model:
self.classes->forAll(c |
 (c.oclIsKindOf(ExecutableNode) and
        c.stereotype.name -> includes("ActivityOperation")) or
 (c.oclIsKindOf(ExecutableNode) and
        c.stereotype.name -> includes("WebService")))
```

Fig. 6. Service composition metamodel

3 Model Transformation for service-oriented web applications development

As mentioned before, the proposed method for service-oriented web applications development is based on the definition of models at different abstraction levels, the basis of the model-driven development paradigm [2], [13]. In the previous section we have defined the metamodels (consequently the models) that must be considered in our method, thus, according to MDA principles, the only issue that must be faced in order to complete the proposal is the definition of the mapping between these models. This process stands for model transformation [11], [14].

3.1 Mapping Rules

Figure 7 shows the modeling process proposed for service-oriented web applications development that includes the models defined in the previous subsections. As stated earlier, in this work we focus on the mapping rules between PIMs, remarked in Figure 7. At PIM level, the process starts by building the *business services model* and includes two intermediate models to finally obtain the *services composition model*.

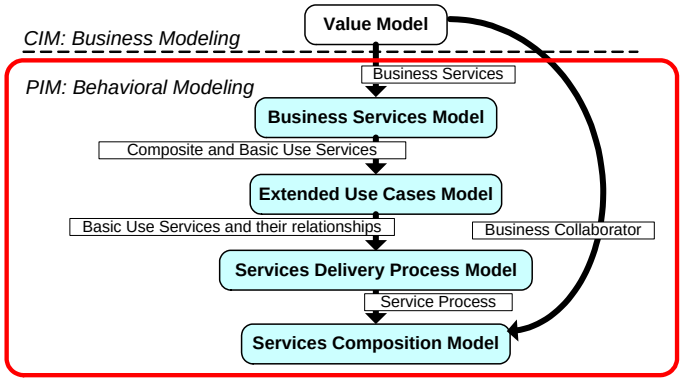


Fig. 7. Modeling process for service-oriented web applications development

In relation to the way mappings should be defined in [11] it is stated that "the mapping description may be in natural language, an algorithm in an action language, or a model in a mapping language". In this case, and as a first approach, we have decided to describe the transformation rules between models in natural language for later expressing them as graph transformation rules. These transformations rules are collected in Table 1. According to [11], as some of the mapping rules of the transformation process require design decisions, it is not possible to automate them completely. As a result, we have made the distinction between the mapping rules that can be Completely (C) or Partially (P) automated.

Table1. Mapping rules between PIMs in the method for service-oriented web applications development

From	To	Mapping Rules	Grade of Autom.
Business Services Model	Extended Use Cases Model	1. Every Service found in the business service model will be split into one or more CompositeUseService (CS) and/or BasicUseServices (BS).	P
		2. Every CS generated will be split into one or more BS.	P
Extended Use Cases Model	Service Delivery Process Model	3. For every BS corresponding to a same BusinessService, there will be a ServiceActivity (SAct) in the service delivery process model that describe this BusinessService.	C
		4. Every extend association identified in the extended use cases model will be represented in the service delivery process model by a ForkNode. The SAct corresponding to the source BS of the extend association must be previ-	C

		ous to the SAct corresponding to the target BS of the extend association.	
		4.1 If the extend association has only one source BS, the fork will present the SAct as an alternative to another flow with no SAct. Later, both flows will meet.	C
		4.2 If the extend association has several sources BS, the fork will present the different SAct as mutual alternatives to another flow with no SAct. Later, all these flows will meet.	C
		5. Whenever a include association is found in the extended use cases model, the SAct corresponding to the source BS of the include association must be subsequent to the SAct corresponding to the target BS of the include association.	C
		5.1 If the include association has several targets, the designer must decide the appropriate sequence for the different SAct corresponding to the target BS (that will be obviously previous to the SAct corresponding to the source BS).	P
Services Delivery Process Model	Service Composition Model	6. Every SAct found in the service delivery process model will be split into one or more ActivityOperation (ActOp).	P
		7. The control flow between ActOps is the same as the flow between their relative SActs.	C
		7.1 In the case of a SAct containing two or more ActOps, the designer has to choose the particular control flow between the ActOps.	P

3.2 Graph Transformation

To observe the MDA principles, the model to model transformation of our method for service composition modeling development must be automated, at least in some extent. To achieve this objective we have decided to use a graph transformation approach [1], [4], [16]. Using a graph transformation approach results in two main advantages: on the one hand, graph grammars are based on a solid mathematical theory and therefore they present a number of attractive theoretical properties that allows formalizing model transformations; on the other hand, the use of graph grammars for mappings definition could be shown as a direct step towards to implementation since projects like Attributed Graph Grammar System (AGG)[15], VIATRA[3] or ATOM3[6] will provide us with the facilities to automate model to model transformations defined as graph transformations. Moreover, as previously mentioned, the term Graph Transformation is used to refer to a particular category of rule-based transformations that are typically represented diagrammatically. So, given that we have already formally defined the mappings in a set of rules, it seems appropriate to translate these rules to graph transformations rules. Finally, from a pure mathematical point of view, we can think on UML-like models as graphs. A graph has nodes and arcs, while an UML model have classes and associations between those classes; this way the fact that models are well represented as graphs is particularly appealing to shorten the

distance between modelers and model transformation developers, a big problem around model transformation. Rule-based transformations with a visual notation may close the semantic gap between the user's perspective of the UML and the implementation of transformations.

To express model transformations by graph grammars, a set of graph rules must be defined. These rules follow the structure LHS:= RHS (Left Hand Side:= Right Hand Side). Both, the LHS and the RHS are graphs: the LHS is the graph to match while the RHS is the replacement graph. If a match is found on the source model, then it is replaced by the RHS in the target model. In this work we have used the approach already applied in previous works like [18] to define the graph rules that collects the transformation rules proposed in Table 1.

According to these guidelines, we have defined the graph rules for the model transformations needed in our proposal for service-oriented web applications development that were susceptible of being expressed by graph grammars.

From now on we present these graph rules next to the respective definition rules in natural language. Figure 8 describes the mapping rules corresponding to transformations from the business services model to the extended use cases model. Figure 9 to 12 describe the mapping rules corresponding to transformations from the extended use cases model to the service delivery process model. Finally, Figure 13 describes the mapping rules corresponding to transformations from the service delivery process model to the service composition model. For the sake of space we have had to reduce the size of these pictures, in some cases they could result difficult to read. In order to improve their clarity, they can be acceded in http://kybele.es/models/MTsowa.htm.

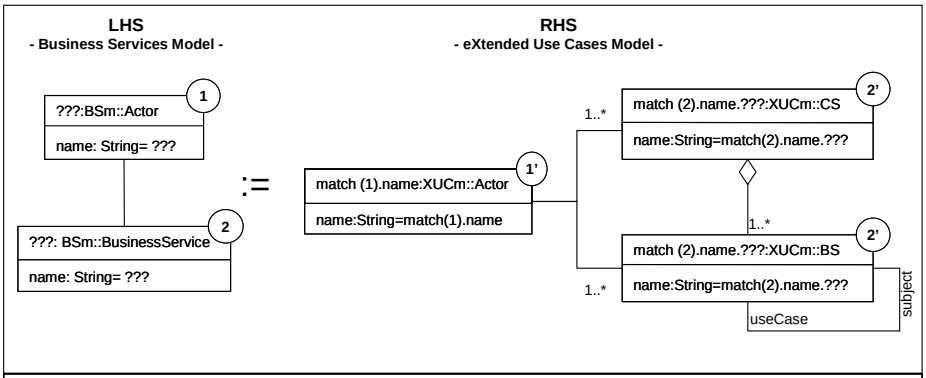
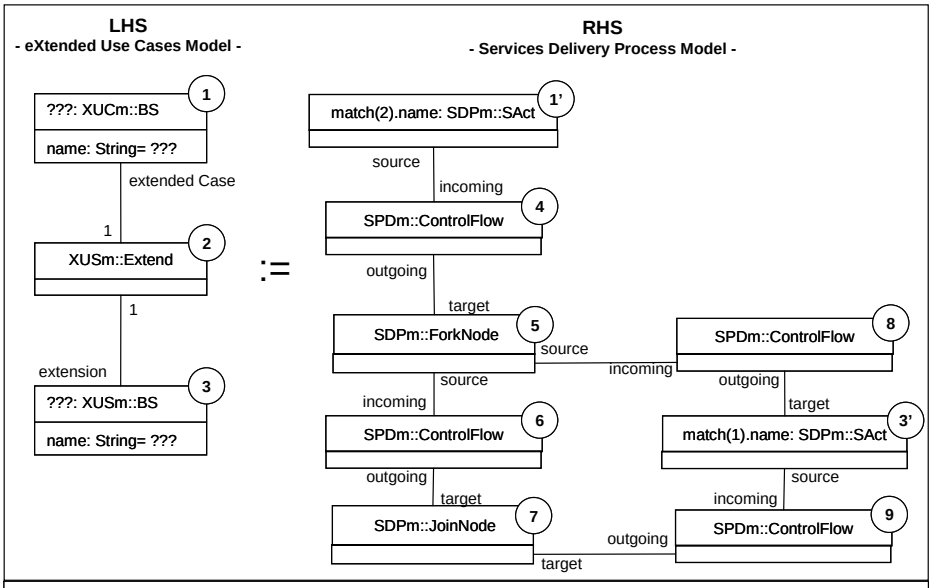


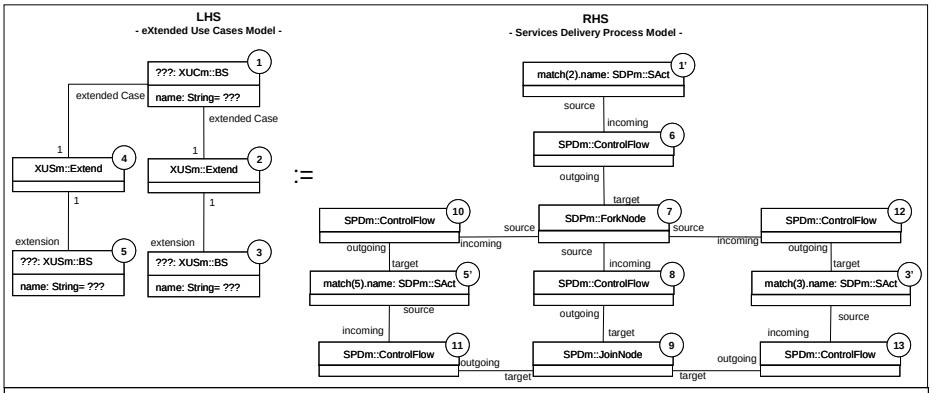
Fig. 8. BusinessServices and Actors in the business services model mapped to CompositeUseServices, BasicUseServices and actors in the extended use cases model



4. Every extend association identified in the extended use cases model will be represented in the service delivery process model by a ForkNode. The SAct corresponding to the source BS (①) of the extend association must be previous to the SAct corresponding to the target BS of the extend association (③).

4.1 If the extend association has only one source BS, the fork will present the SAct as an alternative to another flow with no SAct. Later, both flows will meet.

Fig. 9. Extend associations in the extended use cases model mapped to the service delivery process model.



4. Every extend association identified in the extended use cases model will be represented in the service delivery process model by a ForkNode. The SAct corresponding to the source BS (①) of the extend association must be previous to the SAct corresponding to the target BS of the extend association (③ y ⑤).

4.2 If the extend association has several sources BS, the fork will present the different SAct as mutual alternatives to another flow with no SAct. Later, all these flows will meet.

Fig. 10. Extend associations (with several sources BasicUseServices) in the extended use cases model mapped to the services delivery process model.

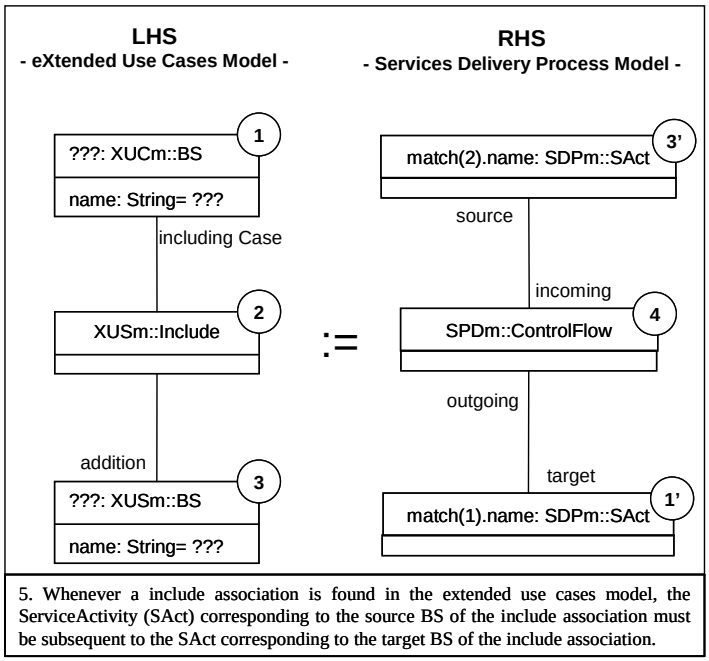


Fig. 11. Include associations in the extended use cases model mapped to the services delivery process model

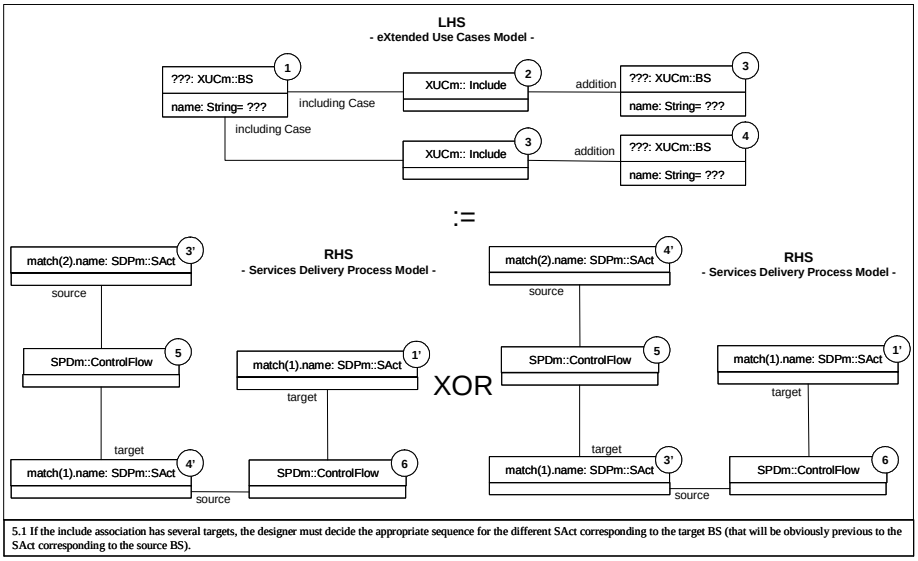


Fig. 12. Include associations (with several target BasicUseServices) in the extended use cases model mapped to the services delivery process model.

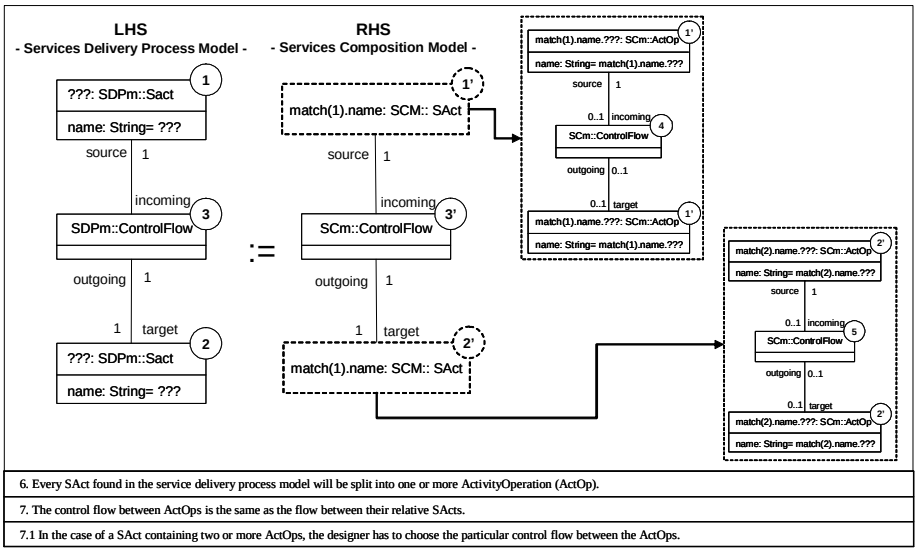


Fig. 13. ServiceActivities in the services delivery process model mapped to the services composition model.

4 Conclusions and Future Works

In this work we have presented the model to model transformations needed to complete an MDA approach for service-oriented web applications development. This way, we have firstly described the metamodels for the PIMs considered by the method. They provide with new elements for service-oriented web applications modeling and extend the behavioral modeling elements of UML 2.0. Next we have defined the mapping rules between these PIMs following a graph transformation approach. As a first approach to model transformations from the proposal for service-oriented web application development, we have firstly defined the transformation rules in a declarative manner for later formalize them with graph rules in order to automate them using some of the existing facilities to automate graph transformations. The mapping rules defined in this work allows obtaining a service composition model that can be easily translate to a specific web service technology, starting form a high level use cases model in which the services required by the web consumers were represented.

This work serves as a clear example of the value of model transformations in Software development: the model to model transformations presented in this work complete the definition of our process for service-oriented web applications development, a contrasted and published method that founds in model transformations the piece that remained to become a completely feasible methodology.

At the present time we are working in the integration of the method described in this work in a CASE tool which is now under development in our research group and

which its early functionalities have already been presented in previous works [17]. Besides, the open issue of making automatic the graph transformations by using existing technologies like ATOM3 is been tackled.

References

1. Baresi, L., Heckel, R.: Tutorial Introduction to Graph Transformation: A Software Engineering Perspective. In Corradini, A., Ehrig, H., Kreowski, H., Rozenberg, G. (eds.): Proceedings of the First international Conference on Graph Transformation. Lecture Notes in Computer Science, Vol. 2505. Springer-Verlag, (2002) 402-429.
2. Bézivin, J.: In search of a Basic Principle for Model Driven Engineering, Novatica/Upgrade Vol. 5, N° 2 (2004) 21-24.
3. G. Csertan, G. Huszerl, I. Majzik, Z. Pap, A. Pataricza and D. Varro, VIATRA — Visual Automated Transformations for Formal Verification and Validation of UML Models, in: Proc. of 17th IEEE International Conference on Automated Software Engineering (ASE'02), IEEE Computer Society, Los Alamitos, CA, USA, 2002, pp. 267-285.
4. Czarnecki, K., Helsen, S.: Classification of model transformation approaches. In: Bettin, J., Emde Boas, G., Agrawal, A., Willink, E., Bezivin, J. (eds): Second Workshop on Generative Techniques in the context of Model Driven Architecture (2003).
5. De Castro, V., Marcos, E., López-Sanz M.: A Model Driven Method for Service Composition Modeling: A Case Study. Int. Journal of Web Engineering and Technology. 2006 - Vol. 2, No.4 pp. 335 - 353.
6. J. De Lara, H. Vangheluwe and M. Alfonseca, Meta-Modelling and Graph Grammars for Multi-Paradigm Modelling in AToM3, Software and Systems Modelling, Vol 3(3), Springer-Verlag. August 2004, pp.: 194-209.
7. Gordijn, J., Akkermans, J.M.: Value based requirements engineering: exploring innovative e-commerce idea. Requirements Engineering Journal Vol. 8, Nº 2 (2003) 114 -134.
8. Harmon, P.: The OMG's Model Driven Architecture and BPM. Newsletter of Business Process Trends (May 2004). Accessible in: http://www.bptrends.com/publications.cfm.
9. Kleppe, A., Warmer, J., Bast, W.: MDA Explained, the Model Driven Architecture: Practice and Promise. Addison Wesley (2003).
10. OMG. UML Superstructure 2.0. OMG Adopted Specification ptc/03-08-02 (2002). Accessible in: http://www.uml.org/.
11. OMG. MDA Guide V1.0.1. Miller, J., Mukerji, J. (eds.) Document Nº omg/2003-06-01 (2001). Accessible in: http://www.omg.org/cgi-bin/doc?omg/03-06-01.
12. Papazoglou, M.P., Georgakopoulos, D.: Serviced-Oriented Computing. Communications of ACM Vol. 46, Nº 10 (2003) 25-28.
13. Selic, B.: The pragmatics of Model-Driven development. IEEE Software Vol. 20, Nº 5 (2003) 19-25.
14. Sendall, S., Kozaczynski, W.: Model Transformation–the Heart and Soul of Model-Driven Software Development, IEEE Software archive Vol. 20, Nº 5 (2003) 42-45.
15. Taentzer, G.: AGG: A Tool environment for Algebraic Graph Transformation. In: Nagl, M., Schürr, A., Münch, M. (eds.): Applications of Graph Transformations with Industrial Relevance. Lecture Notes in Computer Science, Vol. 1779. Springer-Verlag, (2000) 481-488.
16. Tratt, L.: Model transformations and tool integration. Software and Systems Modeling, Vol. 4, Nº 2 (2005), 112-122.
17. Vara, J.M., De Castro, V. Marcos, E.: WSDL automatic generation from UML models in a MDA framework. International Journal of Web Services Practices Vol. 1 (2005) 1-12.

18.Vara, J. M., Vela, B., Cavero, J. M., y Marcos, E. Model Transformation for Obtect-Relational Database development. ACM Symposium on Applied Computing 2007 (SAC 2007). Seul (Korea), March, 2007
19.Verner, L.: BPM: The Promise and the Challenge. Queue of ACM Vol. 2, Nº 4 (2004) 82-91.

Appendix A: Stereotypes of MIDAS/BM profile

This appendix includes all the stereotypes defined in the MIDAS/BM profile. It defines the new modeling elements which extend the existing metaclasses of the UML metamodel. For each modeling element we describe UML metaclass extended, semantics and notation.

Business Services Model	
"BusinessService"	
Extend	UML metaclass "useCase"
Semantics	Represent a complex functionality, offered by the system, which satisfies a specific need of a consumer.
Notation	<<*BusService*>>
Extended Use Cases Model	
"CompositeUseService"	
Extend	UML metaclass "useCase"
Semantics	Represent a functionality that is required to carry out a business service, which is composed of other basic or composite use services.
Notation	<<*CS*>>
"BasicUseService"	
Extend	UML metaclass "useCase"
Semantics	Represent a functionality that is required to carry out a business service
Notation	<<*BS*>>
Services Delivery Process Model	
"ServiceActivity"	
Extend	UML metaclass "ActivityNode"
Semantics	Represent a behavior that is part of the execution flow of a business service.
Notation	<<*SAc*>>
Services Composition Model	
"ActivityOperation"	
Extend	UML metaclass "ExecutableNode"
Semantics	Represent an action that is supported by a service activity.
Notation	<<*AOp*>>
"WebService"	
Extend	UML metaclass "ExecutableNode"
Semantics	Represent an action that is supported by a service activity which can be implemented by means of a web service.
Notation	<<*WS*>>