Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 194 (2008) 23–37
www.elsevier.com/locate/entcs

Modelling of Service-Oriented Architectures with UML

Marcos López-Sanz[a,1,2] César J. Acuña[a,1,3]
Carlos E. Cuesta[a,1,4] Esperanza Marcos[a,1,5]

[a] *Dpto. de Lenguajes y Sistemas Informáticos II*
Escuela Técnica Superior de Ingeniería Informática
Universidad Rey Juan Carlos
C/Tulipán s/n 28933 - Móstoles (Madrid), Spain

Abstract

Nowadays, service-oriented architectures are becoming gradually more important. The vast diversity of implementation and support platforms for this kind of architectures (such as Web, Grid or even CORBA) increases the complexity of the development process of service-based systems. With the aim of facilitating the development of service oriented solutions, we propose the specification of an architecture centric model driven development method. To achieve this, we study the architectural properties of the SOA paradigm and follow a development approach based on the MDA proposal. MDA proposes a separation of the development process in abstraction levels. This makes MDA suitable to tackle the development of service-oriented systems. This paper describes a UML profile for the PIM-level service-oriented architectural modelling, as well as its corresponding metamodel. PIM (Platform Independent Model) level is chosen because it does not reflect constraints about any specific platform or implementation technology. To exemplify and validate the profile, a case study is presented in which the proposed profile is used.

Keywords: Service-Oriented Architectures, Model-Driven Architecture, PIM-level modelling, UML Profiles

1 Introduction

In the last years the development of systems based on services has grown in importance. The introduction of the service orientation principles into the business field [11], the specification of new technology standards for services [19] or the use of

[1] This research is partially granted by project GOLD (TIN2005-00010) financed by the Ministry of Science and Technology of Spain, the FoMDAs project (URJC-CM-2006-CET-0387) cofinanced by the Rey Juan Carlos university and the Regional Government of Madrid, and the Spanish project "Agreement Technologies" (CONSOLIDER CSD2007-0022, INGENIO 2010).

[2] Email: marcos.lopez@urjc.es

[3] Email: cesar.acuna@urjc.es

[4] Email: carlos.cuesta@urjc.es

[5] Email: esperanza.marcos@urjc.es

doi:10.1016/j.entcs.2008.03.097

services as basis in the construction of middleware systems [8] are among the main motivations. However, several problems have come up along with this evolution. Issues like the migration of execution platform, the design and implementation of intricate lifecycles, the increase in the complexity of the development process and the lack of a precise definition of the concepts involved in SOA (Service Oriented Architecture) solutions are among them. So, different actions should be taken to tackle these problems. We focus in two:

(i) To accomplish a study of the architectural principles governing the SOA designs. The architecture reflects the structure and behaviour of a system and how it evolves as time elapses. Moreover, the architecture of a service-oriented system should include features related to business processes and organizational aspects. The integration of business models, which has lead to its quick evolution and spreading, is one of the main benefits of SOA.

(ii) To follow a methodological approach to reduce the complexity of the SOA development process. One of the current trends that more importance is gaining is the model-driven approach. The ideas behind the MDA (Model-Driven Architecture)[16] proposal can facilitate and improve the development of SOA solutions.

Consequently, to solve the problems stated at the beginning, a SOA development method (SOD-M) based on the MDA principles and architecture-centric could be used. MDA conceives models as first class elements during system design and implementation and establishes a separation of the development process in three abstraction levels, namely CIM, PIM and PSM. Its main feature, however, is the definition of mappings between the models defined in each level and between levels, what makes possible the automation of the development process. In our case, the methodological framework in which we lean on is MIDAS, a methodological framework for the model-driven development of Web Information Systems (WIS) [5].

In previous research works [14], we made an in-deep study of the convenience of extending the MDA proposal to support the specification of the architectural modelling aspect within a MDD (Model Driven Development) process. Following its conclusions, in this article we present a UML profile for the PIM-level service-oriented architectural modelling, together with the correspondent metamodel. To illustrate and validate it we use, as case study, an extension for a Web Information System for medical image management called GESiMED [1].

There are some other works related with the topic of this article. They also deal with the definition of the SOA principles ([2], [13], [17]), with UML profiles for service-based developments ([3], [10], [20]) and even the MDA principles applied to SOA ([21]). They are analyzed in detail in Section 4.

The remainder of the article is structured as follows: Section 2 gives a general overview of the MDA framework in which the research is framed. Section 3 presents, firstly the concepts involved in the UML profile by means of depicting the associated meta-model and, secondly, UML profile for the PIM-level service-oriented

architectural modelling together with an illustrative case study. After analyzing some related works in Section 4, Section 5 presents the main conclusions and future works of our research.

2 MDA Research Framework

The research work presented in this article is part of MIDAS, a methodological framework for the development of WIS based on MDA. MIDAS proposes a model-driven architecture supported by three orthogonal dimensions (see Figure 1).

- *Vertical Axis (Y)*: This axis reflects the three abstraction levels of the original MDA proposal: CIM (Computation Independent Models), PIM (Platform Independent Models) and PSM (Platform Specific Models).
- *Horizontal Axis (X)*: Models are separated in the different concerns involved in the WIS development: Content, Hypertext and Behaviour. As can be seen in figure 1 the separation of aspects only affects to the PIM and PSM levels because CIM level just focuses on domain and business models.
- *Traversal Axis (Z)*: Models in this axis are referred to aspects that have influence on other models of the cross-cutting axis. Here is where the architecture model should be defined as well as other models such as the semantic model.

As can be seen in Figure 1, the architectural aspect affects both the PIM and PSM levels of the model architecture. This is because MIDAS follows an ACMDA (*Architecture Centric Model-Driven Architecture*) approach since it defines a method for the development of WIS based on models and guided by the architecture. The architecture is considered to be the driving aspect of the development process. It allows to specify which aspects and models in each level are needed according to the software architecture design.

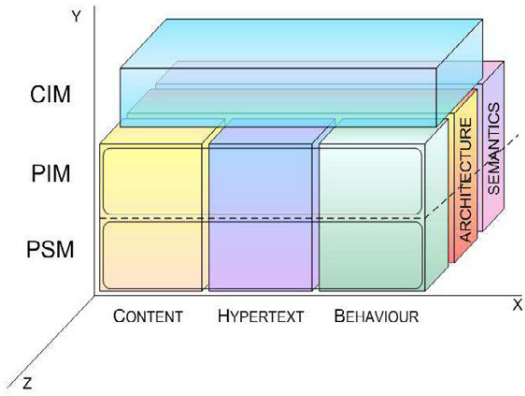

Fig. 1. MIDAS model architecture

The architectural models are divided into two levels which correspond to the PIM and PSM levels. The reason to make this division is that, with an architec-

tural view of the system at PIM-level, we ensure that there are no technology or implementation constraints in the model besides facilitating the possibility of establishing different PSM-level models according to the specific target platform and derived from one unique PIM-model.

Moreover, and applying this conception to a system following the SOC paradigm, the services that appear at the PIM-level architecture model will be understood as conceptual services, classified depending on the role or functionality given and the entities that group those services. Of course, the dependencies and limitations among the services, entities and other components will be modelled without reflecting, in any case, any implementation or platform feature or restriction.

At the PSM level, however, the influence of the platform or specific technology over the PIM-level architectural design should be modelled. The elements to appear will be the same as in the PIM level but refined and applied over a specific service implementation technology. Nevertheless, this topic is beyond the scope of this article.

3 Proposal of PIM-Level UML Profile for SOA

In this section we introduce our proposal of UML profile to be used in the definition of the architecture model for SOA at PIM-level. First we describe the concepts in the UML profile using the associated metamodel and then we present the UML profile itself illustrated with a case study.

3.1 Concepts involved in the profile definition

The set of concepts that appear in the PIM-level architecture metamodel depicted in Figure 2 are explained next:

3.1.1 Service providers

A Service-Oriented Architecture is built upon independent entities which provide and manage services. Because SOA is widely used as a way to integrate enterprise applications, the existence of these providers is justified by the necessity to project the business organizations involved in those processes into the architecture model. These providers act as service containers in charge of presenting the services contained to the world. And also, as it will be explained later, in charge of managing user interaction.

In the presented architecture model these entities are identified as UML 2.0 packages. They are not considered as subsystems because, so far, they do not add or modify neither the inner services nor the dependencies or relations between them.

Service providers can be classified into two main groups: inner service providers (*innerProvider* in the metamodel), which are internal to the system designed (usually the system itself) and outer service providers (*outerProvider* in the metamodel), which are identified as the external entities containing services which collaborate with the system to perform a specific task of value for the system but which are not

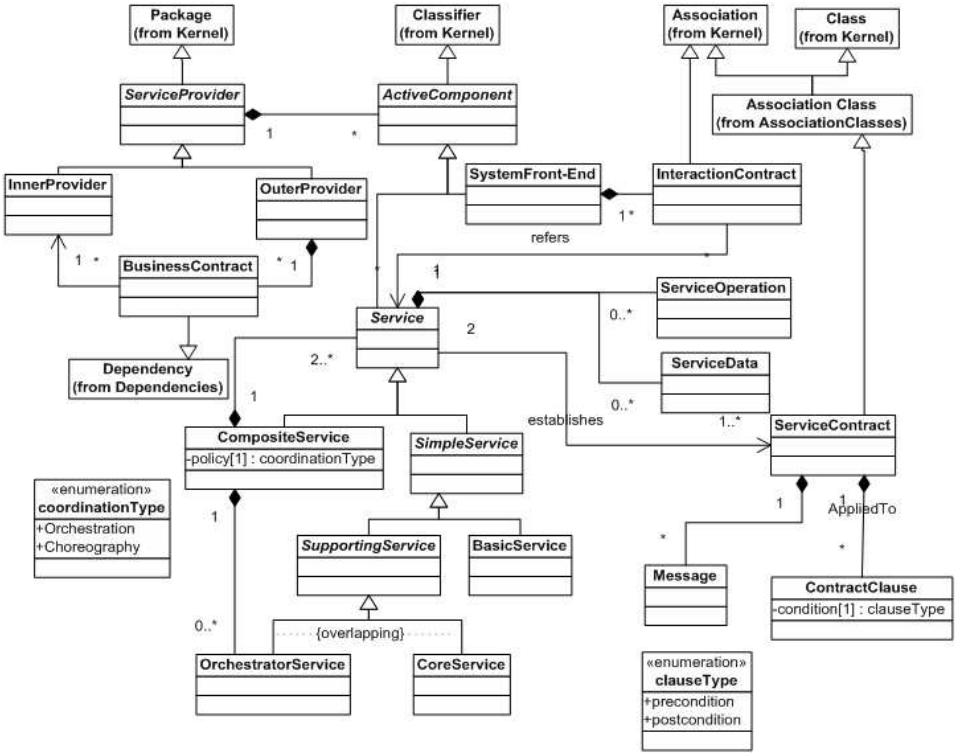

Fig. 2. PIM-level SOA Metamodel

under its control.

The relationship between service providers is modelled as a dependency understood as a "business contract" (*bussinessContract* in the metamodel) that represents a sort of communication between the services belonging to the providers.

This element can be found in the bibliography under several other names (with slight differences in concept or role) such as service manager [11], container [2], market-maker, service aggregator or service provider [17], context [21] or service owner [6].

3.1.2 Active components

In a SOA-based solution, the main elements are services; however, as user responses are needed in many cases, a first-class element which retrieves the user interaction must be represented in the architecture model, namely *SystemFront-End*. Both elements, services and interaction components, are grouped under a common element named *ActiveComponent* which is controlled by a specific *ServiceProvider*.

System Front-End

By definition, services lack of the ability to interact or, in particular, to modify its behaviour according to user interaction. Additionally, it is the user, through the application front-end, who usually initiates the business process implemented in the system. Those reasons justify the necessity to include a component in

the architecture model able to represent those capabilities. *SystemFront-End* components will be, therefore, identified as system boundaries.

Service

The main role of services inside SOA is to support the functionalities offered by the system. Our vision of service (at PIM level) is aligned with that of the OASIS reference model for services:

Definition 3.1 A *service* is a mechanism to enable access to a set of one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description [15].

Services inside a SOA-based system play different roles. They can be classified according to its atomicity in simple services (*SimpleService* in the metamodel) or composite services (*CompositeService* in the metamodel). In this point we have to differentiate service providers from composite services. Since both are means of grouping services, composite services are distinguished for having a property indicating the coordination policy used as well as a well-known interface (because of being a kind of services). This property (named policy) can be set to two possible values: Orchestration o Choreography. In case of taking the Orchestration value, it is compulsory that at least one of the compound services assume the coordinator role.

Simple services, in turn, can be divided into basic services (*BasicService*) or supporting services (*SupportingService*). Basic services represent those services with distinctive functional meaning that typically encapsulates a high-level business concept. Supporting or specialized services perform actions clearly identified and not necessarily with direct relation with the modelled system functionality.

These supporting services can be again classified in two: orchestration services (*OrchestrationService*) or core services (*CoreService*). A service will assume the "orchestrator" role in case of being responsible of performing coordination tasks inside a composite service, that is, for the execution cycle involving the invocation of several services. Core services, on the other hand, perform tasks needed for other services to work correctly. Among the last ones we could find, for example, location services, registry services, security management services, etc. By identifying different service levels (core and others) at PIM-level we favour the automation process advocated by the MDA approach as it will be easier to separate them when describing what services belong to the platform or to upper implementation levels.

Service Description

Despite the type of service modelled, all of them have a set of operations (*serviceOperations*) and data stored (*serviceData*). Although many of the references in the bibliography understand services as stateless entities, this is an aspect that depends mostly on the service implementation technology and so, at PIM-level, we think that state data should be modelled (as part of the service data in the service description). Additionally, operations offered by the services should be included in

the service description. We identify service operations as atomic functionalities that collaborate to outline the service features.

Other reason to model data as an element inside the architecture model is that, from the point of view of our development methodology, these data represent a link point between the parts of the methodology in charge of modelling the data aspect (Content aspect in the MIDAS framework) and the architecture model guiding the development process.

3.1.3 Service Interactions and Messages

Services relate, communicate and interact with each other through contracts. The contracts in a service oriented architecture model act as connectors between the different architecture components. These contracts will have different features depending on the components connected. So, we refer to *ServiceContract* when talking about service-to-service connections and *InteractionContract* when talking about the communication between a *SystemFront-End* and a service.

Service contracts

Communication between two services relies on an asymmetric interaction based on request/reply message exchange. Contracts established between services must reflect the services that are involved in the contract, the roles played, and other several properties that also define the interaction, for example, purpose, constraints or contract expiration time. The conditions under which the contract takes place refer, usually, to non-functional requirements such as quality of service, security, interaction protocol or message order, type of communication (synchronous or asynchronous), etc.

The operations that take part of the contract established between services can be understood as the roles played by the contractors. Other sources like Krafzig et al. [11] consider the service contract as part of the service description itself. Although the way to interact with a service depends mainly on the functions offered by a service and shown in its interface, we think that, by making the service contract an independent element of the architecture model, we ensure the independence of the different relations a service can establish with different services, each of them with its own restrictions and characteristics. Other reason that supports our choice is that, in future refinements of the model, the service providers could create specific contract restrictions or templates unfeasible to be applied if modelled inside the services.

Contract Clauses

It is possible that for two services to communicate they fulfil some pre- or post-conditions. In our architecture model this clauses are represented as restrictions associated to the *serviceContract* element and using OCL as description language. So, OCL is used to define the dynamicity of the architecture as the interactions between the components of the architecture will take place only if the conditions are satisfied. However, and so far, the constraints are not dynamic themselves.

Interaction with the System Front-End element
The communication between the application front-end and the services may be as simple as a parameterized invocation of the service from the front-end component, or may be more complex, for example, based on a negotiation protocol in which all the interaction clauses are set. In the metamodel this concept is represented by the *InteractionContract* element.

Messages
Messages in SOA represent the communication item exchanged between services. Each message (or, more properly, message type) is related to a service contract. The most important feature of messages is that they have to be understood by both the emitter and receiver services, so the attributes of this concept should include references to the ontology model or semantic constraints involved in the message.

Although the semantics associated to the message appear in the architecture model, the message format does not. The format of a message is an issue that depends mostly on the implementation technology and so it should be modelled in the correspondent PSM-level models. However, although exchanged messages are inherently related to the operations offered by every service, we consider that, by identifying the messages as separated entities, the model is more flexible to support the evolution of the architecture. This approach also appears in other works such as Wada et al. [20].

3.2 PIM-Level UML Profile for SOA

In this section we show the main stereotypes needed to represent the concepts described previously in a UML model for SOA. Figure 3 shows the main components of the profile created. Afterwards, the use of each stereotype by means of a case study is illustrated.

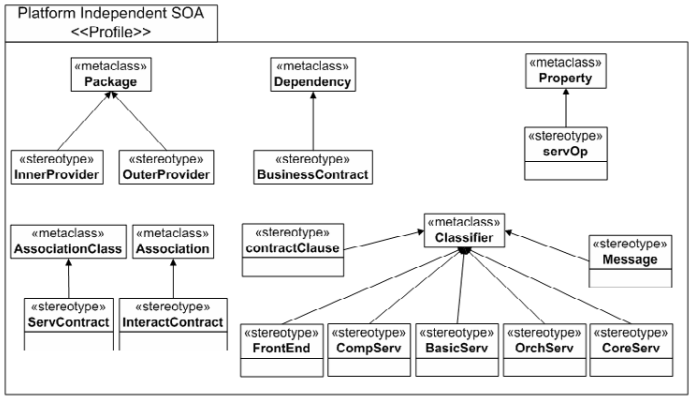

Fig. 3. UML profile for SOA

Table 1 lists the stereotypes defined as well as the base UML metaclass from which they derive and the tag used to represent each stereotype.

NOTATION	RELATED CONCEPT	BASE UML META-CLASS	MEANING
`<<innerProvider>>`	Inner provider	Package	Provider of inner services
`<<outerProvider>>`	Outer provider	Package	External provider of services
`<<businessContract>>`	Business contract	Dependency	Business contract established between providers
`<<FrontEnd>>`	System Front-End	Classifier	Component that interact with user
`<<CompServ>>`	Composite service	Classifier	Service compound of services
`<<BasicServ>>`	Basic service	Classifier	Represents a basic funcionality
`<<OrchServ>>`	Orchestrator service	Classifier	Acts as orchestrator in a coreography
`<<CoreServ>>`	Core service	Classifier	Offers a supporting functionality to other services
`<<ServContract>>`	Service contract	Association Class	Acts as connector of services
`<<InteractContract>>`	Interaction contract	Association	Connector between services and system front-end
`<<contractClause>>`	Restriction clause	Classifier	Indicates a pre- or post-condition to be fulfilled
`<<servOp>>`	Service operation	Property	Atomic functionality of the service
`<<Message>>`	Message	Classifier	Exchanged information token

Table 1
Stereotypes for the UML Profile

The case study used as example to apply and validate the defined SOA UML profile comes up from the necessity of extending the features offered by GESiMED, a WIS for the management of medical images [1]. In particular, we wanted to add to this system the capability of performing, in a unique process, the acquisition of images from a database and its subsequent processing in order to be displayed on a Web page. Moreover, this process requires the payment of a processing fee from the user.

From this newly added functionality we define a partial architecture model for the GESiMED system based on the SOA principles and according to the concepts defined in the presented metamodel. The complete architecture model should be completed by adding elements from other system use cases.

3.2.1 Service providers

In our case study, the functionalities defined to complete the task of acquiring and processing a medical image are supported by four different service providers as can be seen in Figure 4.

In Figure 4, service providers are stereotyped with the `<<innerProvider>>` and `<<outerProvider>>` tags, depending on whether they contain and manage services controlled by the system or services provided by external entities (outer providers). The relation between providers is stereotyped with a `<<businessContract>>` tag indicating that there will be a communication between their services. In our case study, we identify as outer providers those who manage the bank services, the

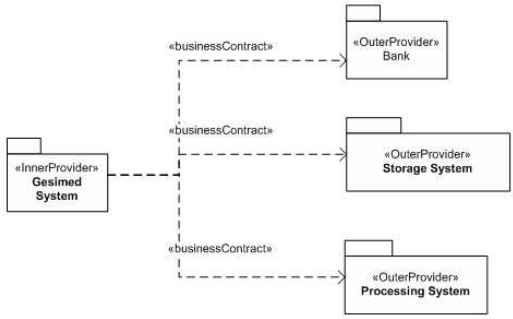

Fig. 4. Relations between providers

storage services and the processing services respectively.

3.2.2 Services and service interactions

As can be seen in Figure 5, we identify, as part of the inner provider, on the one hand, a system front-end (*WebInterface*, stereotyped with `<<FrontEnd>>`) which will be responsible to retrieve user activity and, on the other hand, two services: a *ImageQueryService* (stereotyped with `<<OrchServ>>` as its main function is to serve as orchestrator for the invocations performed to the outer provider services) and *Locator* (stereotyped with `<<CoreServ>>`, its functionality is considered as a supporting facility for other system services).

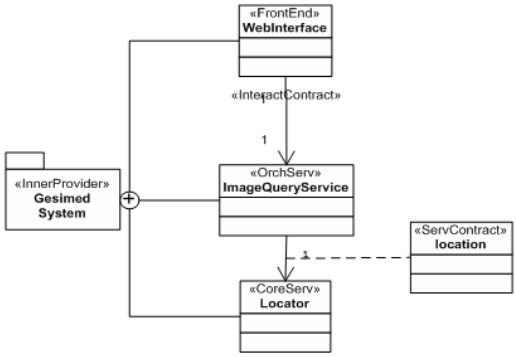

Fig. 5. Relations between a provider and its services

The relation between the *WebInterface* and the *ImageQueryService* (Invocation) is stereotyped with the `<<interactionContract>>` tag. In contrast, the relation between the *ImageQueryService* and the *Locator* is built upon an association class stereotyped with `<<serviceContract>>`.

3.2.3 Service contracts and messages

Service contracts are also established among the *ImageQueryService* and those from the outer providers (*BankService*, *StorageService* and *ProcessingService*). Each service contract has a variety of message that can be used as information tokens in the communication between services. Figure 6 focuses on showing how the stereotypes related to these aspects are used.

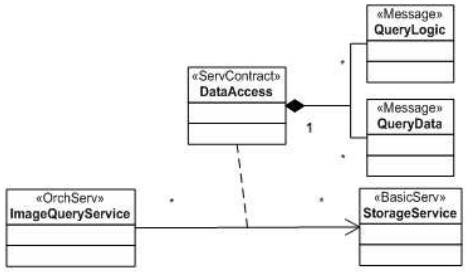

Fig. 6. Relations between services and its associated concepts

The *ImageQueryService* establishes a communication with the *StorageService* through a service contract named *DataAcces*. This contract has two types of messages that can be used while this contract takes place: *QueryLogic* and *QueryData*, stereotyped with `<<Message>>` and understood as the only message types that can be exchanged. This implies that all the operations offered by the services (and used by the contractors during the interaction lifetime) will use only this kind of messages.

Putting all together, figure 7 shows the complete model for the PIM-level architecture for SOA.

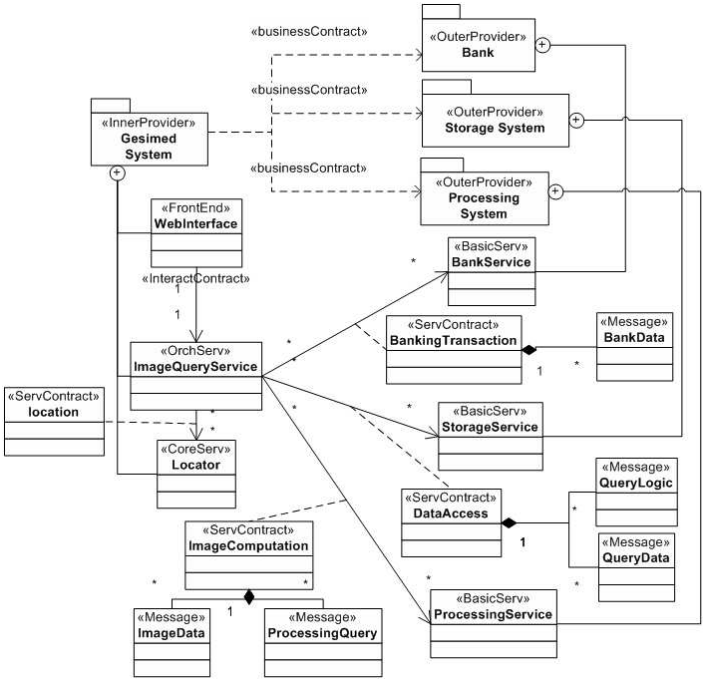

Fig. 7. Complete SOA architecture model of the case study

4 Related Works

The model-driven development of SOA systems have been tackled by several other authors in the last few years.

Baresi et al. [4] propose to model an validate Service-oriented architectures using UML to represent SOA static concepts, graph transformation rules to represent the behaviour and transition systems to validate them. Although they use UML as abstraction method to model the architecture, this proposal considers that any SOA development is built upon service clients, providers and service discovering agents. This vision constraints the scope in which SOA can be used since their proposal cannot be generalized to other execution platforms apart from the ones that follow this scheme.

Starting from the proposal of Baresi et al., Heckel et al. [9] define a UML profile for SOA modelling but taking into account the MDA principles, that is, the separation of PIM and PSM levels. Their UML profile only allows to define two kinds of services (provider and registry) and does not include any facility to model semantic or constraint issues (for example applied to exchanged messages).

Other work related to the SOA modelling is the one presented by Zhang et al. [22]. They define a component metamodel in which services are considered as first-class elements. Other common points between this proposal and the one presented in this paper are the definition of service contracts, the roles given to the services in the contracts or the identification of pre- and post- conditions. However, they restrict the composition of services merely to choreographies and, most important, they don't deal with the problem of user interaction as they do not define any modelling component to support it.

Wada et al. [20] present a solution based also on UML. They focus on the representation of non-functional aspects and the definition of services, messages, connectors and pipes as the only first-class available elements. Other drawback of this proposal is that all the restrictions are gathered inside comments. This represents an important inconvenient if we want to automate the development process or to validate models.

Amir et al. [3] discuss another proposal of UML profile for SOA modelling. Following the principles of Web services, they define 5 different profiles, each of them associated to a SOA concept: resource, service, message, policy and agent. This proposal is too close to the implementation technology and so it wouldn't be suitable to develop a generic SOA system where a platform exchange should happen.

There are other proposals coming from the enterprise research field. Papazoglou et al. [17], for example, makes a classification of the concepts involved in SOA development but from the viewpoint of the role given to each component. We consider this viewpoint by including a hierarchy of services in our metamodel. Moreover, their research is based on the Web Services principles. This circumstance appears also in other works from the enterprise research field such as the ones by Aiello et al. [2], Lublinsky et al. [13], Erl [7], o Krafzig et al. [11]. This is mainly due to the fact that they start from the definition of service (and SOA by extension) given by

the W3C, where a service is defined as a stateless distributed entity. This represents a strong restriction when trying to define the system architecture at a conceptual and platform independent level as in the PIM level of MDA.

The UML 2.0 profile created by IBM [10] is one of the closest proposals to the content presented in this article. On it a generic model with the main concepts of the SOA paradigm is presented from a more conceptual viewpoint and not so constrained to the Web Service concepts.

There are other proposals related with the definition of SOA architectures [18] [12] which include the specification of ADLs (Architecture Description Language) for the formal definition of architectures, but it is a topic beyond the scope of this article. The latter (the one by Krüger et al.) defines, additionally, a systematic development process based on sequence diagrams as graphical notation for the description of service architectures.

5 Conclusions and Future Works

The SOC paradigm is gaining importance in the software development field principally because of the broadening of the fields on which it is being used and accepted as guiding paradigm. However, several problems, such as an increase of the complexity development process, are derived from this evolution. This motivates the necessity to design new development methods to assist the development process of SOA-based systems.

In this article we have presented a UML profile for the design of PIM-level SOA-based architecture models. The results of the research work presented in this article are part of MIDAS, a SOD-M and methodology for the development of WIS.

The model-driven approach in general and the MDA proposal in particular have turned out to be a great advance in software development. The separation in abstract levels (CIM, PIM and PSM) and the definition of transformation rules between the models defined within these levels and among levels are the foundations of the MDA proposal. The application of the MDA principles to the SOA paradigm favours the development of solutions based on services. On the other hand, the design of the architecture is a critical aspect in SOA-based developments since one of its main goals is to achieve flexible implementations of interacting software entities.

Although there are other proposals for the service-oriented system development, many of them consider Web Service principles as basis keeping them apart from generic designs. The ones which are enough generic to avoid fixing to a specific technology usually can not be applied to MDA or do not use a high-level standard notation to help the development process. Among the proposals which define UML profiles for SOA, practically none of them have a complete development method associated and only focus on the ways of representing the SOA principles and not in the entire development process. Finally, none of them allows the representation, as first-class architectural element, of a component in charge of interacting with the user.

Introducing a PIM-level architectural model gives an abstract view of the system

which allows to implement the system modelled in different target platforms (by means of different PSM-level architectural models). At this moment we are working on the specification of PSM-level SOA architectural models for different service execution platforms (Web Service, Grid, CORBA, agents, etc). We expect to give a formal representation of the concepts sketched in this article by means of an ADL which allow us to represent the semantics shown in the metamodel as well as its restrictions.

Finally, since this proposal is framed in the MIDAS methodology, we are currently studying the influence of the designed architectural model in some other parts of MIDAS as well as the transformation rules from one model to another.

Acknowledgement

This research is partially granted by project GOLD (TIN2005-00010) financed by the Ministry of Science and Technology of Spain, the FoMDAs project (URJC-CM-2006-CET-0387) cofinanced by the Rey Juan Carlos university and the Regional Government of Madrid, and the Spanish project "Agreement Technologies" (CONSOLIDER CSD2007-0022, INGENIO 2010).

References

[1] Acuña C., Marcos E., de Castro V. and Hernández J.A. A Web Information System for Medical Image Management. *5th International Symposium on Biological and Medical Data Analysis.* Barcelona (Spain). LNCS 3337. Springer-Verlag. pp. 49-59. November 2004.

[2] Aiello M. and Dustdar S. (2006). Service Oriented Computing: Service Foundations. *Proc. of the Dagstuhl Seminar 2006.* Service Oriented Computing, vol. 05462. Germany, July 2006.

[3] Amir R. and Zeid A. (2004). An UML Profile for Service Oriented Architectures. *Companion to the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2004.* pp. 192-193

[4] Baresi L., Heckel R., Thone S. and Varro D. Modeling and validation of service-oriented architectures: Application vs. style. *Proc. of the 9th European Software Engineering Conference (ESEC/FSE 2003)* Helsinki, Finland, September 1-5, 2003.

[5] Cáceres, P., Marcos, E. and Vela, B. (2003). A MDA-Based Approach for Web Information System Development. *Workshop in Software Model Engineering (WISME'03).* URL: http://www.metamodel.com/wisme-2003/.

[6] De Roure D. et al. *The Semantic Grid: A Future e-Science Infrastructure.* International Journal of Concurrency and Computation: Practice and Experience, 5. 2003.

[7] Erl T. (2005). Service-Oriented Architecture: Concepts, Technology and Design. Upper Saddle River: Prentice Hall PTR.

[8] Foster, I. and Kesselman, C. *The Grid: Blueprint for a New Computing Infrastructure.* Ed. Morgan Kauffmann, 1998. ISBN 1558604758.

[9] Heckel R., Küster J., Thöne S. and Voigt H. Towards a UML Profile for Service-Oriented Architectures. *Workshop on Model Driven Architecture: Foundations and Applications (MDAFA '03).* University of Twente, Enschede, June 2003.

[10] Johnston S. *UML profile for software services.* IBM DeveloperWorks, April 2005. URL: http://www-128.ibm.com/developerworks/rational/library/05/419_soa/

[11] Krafzig D., Banke K. and Slama D. (2004). *Enterprise SOA Service Oriented Architecture Best Practices.* Upper Saddle River: Prentice Hall PTR.

[12] Krüger I. H. and Mathew R. Systematic Development and Exploration of Service-Oriented Software Architectures. Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA'04), pp.177-187. Oslo, Norway, June 12-15, 2004.

[13] Lublinsky B. Defining SOA as an architectural style: Align your business model with technology. URL: http://www-128.ibm.com/developerworks/webservices/library/ar-soastyle/index.html. IBM DeveloperWorks site, 09 Jan 2007.

[14] Marcos, E., Acuña, C. J. and Cuesta, C. E. (2006). Integrating Software Architecture into a MDA Framework. *Proc. of the Third European Workshop on Software Architectures (EWSA 2006)*. pp. 127-143. Nantes, France, September 4-5, 2006.

[15] OASIS, 2006. *Reference Model for Service Oriented Architecture. Committee draft 1.0.* http://www.oasis-open.org/committees/download.php/16587/wd-soa-rm-cd1ED.pdf

[16] OMG, 2001. *OMG Model Driven Architecture.* Miller, J., Mukerji, J. (eds.). Document Nro. ormsc/2001-07-01. URL: http://www.omg.com/mda.

[17] Papazoglou M. P. Service-Oriented Computing: Concepts, Characteristics and Directions. *Proc. of the Fourth International Conference on Web Information Systems Engineering (WISE'03)*. pp. 3-12. Roma, Italy, December 10-12, 2003.

[18] Rennie M. W. and Misic V. B. Towards a Service-Based Architecture Description Language. TR 04/08, Technical Report, University of Manitoba, Canada, August 2004.

[19] W3C, 2002. *Web Service Standards.* URL: http://www.w3.org/2002/ws/.

[20] Wada H., Suzuki J. and Oba K. Modeling Non-Functional Aspects in Service Oriented Architecture. *Proc. of the 2006 IEEE International Conference on Service Computing.* Chicago, Illinois, September 2006.

[21] Zdun U. and Dustdar S. (2007). Model-driven integration of process-driven soa models. *Accepted for publication in Invited to the International Journal of Business Process Integration and Management (IJBPIM)*, 2007.

[22] Zhang T., Ying S., Cao S. and Jia X. A Modeling Framework for Service-Oriented Architecture. *Proc. of the Sixth International Conference on Quality Software (QSIC 2006)*, pp. 219-226. Beijing, China, October 27-28, 2006