Using an Architecture-Centric Model-Driven Approach for Developing Service-Oriented Solutions: A Case Study

Marcos López-Sanz, César J. Acuña, Valeria de Castro, Esperanza Marcos, and Carlos E. Cuesta

Department of Computing Languages and Systems II, Rey Juan Carlos University
C/ Tulipán, s/n – 28933 Móstoles, Madrid, Spain
{marcos.lopez, cesar.acuna, valeria.decastro, esperanza.marcos, carlos.cuesta}@urjc.es

Abstract. As services continue achieving more importance in the development of software solutions for the Internet, software developers and researchers turn their attention to strategies based on the SOC (*Service-Oriented Computing*) paradigm. One of these development approaches is SOD-M specifically designed for building service-oriented solutions. In this article we present the results of redesigning a real-world Web-based Information System, called MEDiWIS, using SOD-M. The main goal of the redesigned MEDiWIS system is to support the storage and management of digital medical images and related information by presenting its functionalities as software services. We analyze in detail the main challenges we have found using an ACMDA (Architecture-Centric Model-Driven Architecture) approach to achieve this goal.

Keywords: Service-Oriented Architecture (SOA), Model-Driven Architecture, Architecture-Centric Development, ACMDA.

1 Introduction

In the last years, with the diversification of companies and the emergence of multi-organizational business processes, a new computing paradigm has been rocketed to the first row of attention: *Service-Oriented Computing* (SOC)[1]. The architectural viewpoint of this paradigm, *Service-Oriented Architecture* (SOA)[8], supports the development of dynamic, loosely-coupled and cross-organizational solutions. These features represent the main issues companies are currently demanding, from the technological point of view, to accomplish their goals [14].

However, this shift of direction to the service-oriented approach requires the precise definition of adequate development methods, techniques and methodologies able to overcome the challenges that arise with that change in the way of thinking. To face these issues, two important points should be taken into account: first, the existence of a vast amount of languages and standards supporting SOA [28]; and, secondly, the increasing importance of using approaches allowing for the reflection and implementation of the concepts that come along with the features of new

R. Meersman, P. Herrero, and T. Dillon (Eds.): OTM 2009 Workshops, LNCS 5872, pp. 350–359, 2009.

businesses. Within the software engineering field, the model-driven approaches have resulted of great value in order to tackle this challenges ([3][4][15] etc.).

The *Model-Driven Engineering* (MDE) approach considers models as the basic artefact to be used in system development and advocates the definition of transformation rules to automatically generate the desired software solution [25]. Using this view as starting point it is possible to reason about the concepts describing a system without constraining to the restrictions imposed by any specific language or technology, and, more importantly, to evolve from a high level conception of the system to obtain models closer to the implementation level. Among the initiatives following that approach we can mention *Software Factories [10]*, from Microsoft, or MDA [22], proposed by the OMG consortium. In particular, the MDA approach has gained a lot of attention in the research community. It suggests a separation of models in several abstraction levels: CIM (*Computation Independent Models*), PIM (*Platform Independent Models*) and PSM (*Platform Specific Models*). Additionally, MDA encourages model evolution to be done through the definition of transformation rules at the metamodel level.

In this article we explore the benefits of using together both the MDA approach and SOA in order to redefine and adapt an existing Web Information System (WIS) for the storage and management of digital medical images called MEDiWIS [1]. We focus specially on the architectural aspect since the methodology we use, MIDAS, is considered to follow an ACMDA approach as it will be explained later. This system was originally developed using a traditional 3-layer architecture approach, in which the presentation layer was represented by several Web pages, the persistence layer was implemented by using a commercial XML-DB solution; and the system business logic (behavioural layer) based on the .Net technology. However, the scope where this tool was used continued to broaden, to the extent of requiring either a complete development from scratch, or a redesign of the existing components by upgrading them with new features. The final decision taken was to transform the system into a service-oriented solution which is what we detail in this article.

Additionally, the redesign of the MEDiWIS system allows us to demonstrate the suitability of the combination of MDA and SOA for the development of software systems from a methodological point of view. To achieve this, we have focused on the architectural view of the SOD-M (*Service-Oriented Development Method*) method, as presented in previous work [7]. By transforming MEDiWIS into a service-oriented solution, we prepare the system to be used not only by a human being, but also as a data source for other multi-purpose applications and integration solutions. In addition, we allow the system to be upgraded easily with new features, such as image co-registration, multi-disciplinary image share, security issues, etc.

The structure of the article is as follows: Section 2 gives a detailed description of the initial build of the MEDiWIS system. Section 3 is devoted to explain the redesign of the MEDiWIS system. In that section the SOD-M and associated methodological framework are also briefly explained. Section 4 gathers some of the most relevant related work and, finally, Section 5 puts together the main conclusions and lessons learned from the work presented in this article.

2 The Case of the MEDiWIS System

MEDiWIS is a WIS for the management and processing of medical images through the Web developed at the Rey Juan Carlos University [12]. This WIS was developed to manage the information related to scientific studies in the neuroscience research field. The initial goal of the system was to offer neuroscience researchers historical medical images storage with easy access and the normalized processing and analysis of medical images. Original medical images are usually obtained, directly from clinical instrumentation machines, in two formats: DICOM and Analyze, which are the accepted standards for the exchange of medical image information [16]. As these standards are primarily used in the medical scope, in order to gain interoperability with other systems, the original MEDiWIS was designed focusing on the XML language for the representation of the exchanged information. Although the MEDiWIS system represents a fully-functional software solution, it was developed focusing specially from the content point of view. The aim of the redesign effort puts more attention to the functional aspect maintaining the importance of XML within the new system.

The overall Web architecture, which can be seen in Figure 1, was structured in three layers: presentation, behavioural and persistence layer.

- **Presentation Layer.** This layer comprises a Web user interface in which a module supporting the upload of sets of image files is included. This interface also allows users to store and properly manage and classify the uploaded files.
- **Behavioural Layer.** This layer is composed of five modules: Query Processor, Image Processor, Image Viewer, XML Generator and Data Transformation.

  The *XML Generator* takes DICOM or Analyze files and transforms them into XML documents. In order to improve the modularity and portability of the system, and before inserting them into the database, it is necessary to obtain an intermediate representation of the DICOM and Analyze files. The *Data Transformation* module takes, as input, the intermediate XML files generated by the *XML Generator* module and transforms them into a different XML document adapted to the requirements of the actual database schema. The *Query Processor* is responsible for building the queries matching the user criteria in order to execute them on the DBMS (*Database Management System*), and showing the results of the queries through the Web interface module. The *Image Processor* module, in turn, allows to perform medical image processing, including registration, filtering and statistical analysis. The results and the information of the procedures used for obtaining those results are stored jointly in the database for further browsing or ulterior inspection. Finally, the *Image Viewer* module is the responsible for transferring the results, in the form of parametric mappings superimposed over anatomical images to the presentation layer.
- **Persistence Layer.** This layer allows the storage of all the clinical study images together with the results from the subsequent image processing. The database works as a massive, historical storage repository, which can be queried an accessed through the Internet in ongoing studies and research projects. To support that issue, we used a commercial solution based on the XML-DB principles. From a conceptual model for clinical trials [12], we developed a database XML schema according to the proposal specified in [26].

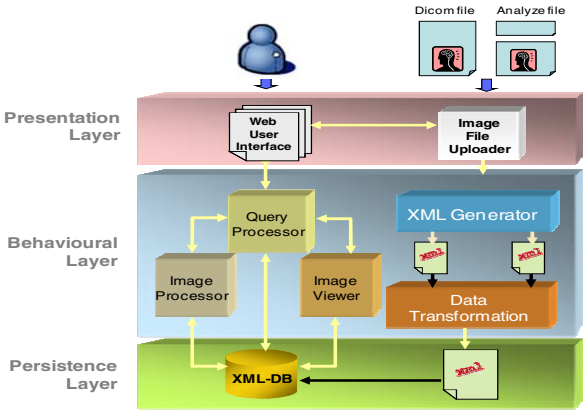

Fig. 1. Architectural Description of MEDiWIS

3 Development of the MEDiWIS System Using an ACMDA Approach

In this section we present the benefits of combining together MDA and SOA. First, we describe the SOD-M method and, second, its application to the redesign of the MEDiWIS system.

3.1 SOD-M and the MIDAS Framework

The research work presented in this article focuses on the architectural modelling aspect of SOD-M, an integrated method for the development of information systems that uses the concept of 'service' as core concept to represent the behaviour of a system. SOD-M is the part of the MIDAS methodological framework [6] devoted to model the behaviour of the system. MIDAS is a methodological framework that proposes the development of information systems following an Architecture-Centric Model-Driven Architecture (ACMDA) approach [19].

As a consequence of this, the architectural aspect affects both the PIM and PSM levels of the model architecture. The architecture is considered to be the driving aspect of the model-driven development process. It allows the specification of which models in each level and which elements inside models are needed according to the software architecture design. The architectural models are divided into two levels corresponding to the PIM and PSM levels. The reason to make this division is that, with an architectural view of the system at PIM-level [17], 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 SOA 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

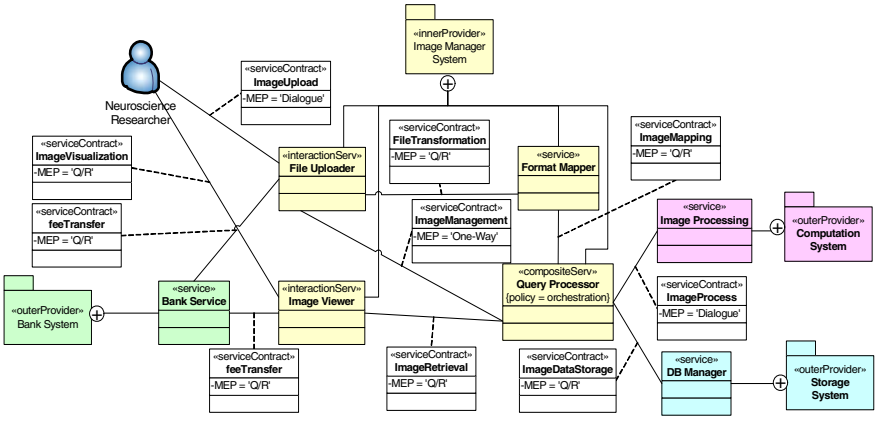

Fig. 2. Service-oriented MEDiWIS: PIM-level architectural model

the services, entities and other components will be modelled without reflecting, in any case, any implementation or platform feature or restriction. The elements to appear at the PSM level will be the same as in the PIM level but refined and applied over a specific service implementation technology.

3.2 Architectural Modelling of the MEDiWIS System

Although the development process includes several other models, ranging from the CIM level to the PSM level, and involving the additional concerns of MIDAS (content, hypertext, etc.), here we will only focus on the architectural models (both PIM and PSM) plus the business level models (CIM models), since they gather the great majority of the changes needed in the new version of MEDiWIS.

First models to be built are those at the CIM level of the MDA proposal. Since these models depict the business and domain environment in which the system is located, both the original and the redesigned version will share the models at that level. For that reason and due to space restrictions we do not show here that model (refer to [7] to see it). In subsequent models of the model architecture, business entities will help to decide the grouping of the services and how responsibilities are assigned to each of them.

Because the system functionality must persist throughout the redesign of the system, MEDiWIS must provide an interface to be used by neuroscience researchers, and therefore it has to cope with the transformation of medical standards to XML. The service-oriented version of MEDiWIS will be comprised of several services in charge of achieving the original functionality. This aspect can be observed in the architectural representation of the system at the platform independent model (PIM) shown in Figure 2. In it, all conceptual services involved in the construction of the system are modelled to build the architectural view of the MEDiWIS system, conforming to the metamodel specified in [17] (service attributes and operations have been removed for the sake of clarity).

Any potential client of the system is represented through the specification of two interaction services. The redesigned system is not only devoted to provide

functionality to human beings (neuroscience researchers) but is also prepared to be used by any other application, which does not necessarily use a Web interface. Consequently, MEDiWIS is now prepared to take part of a future global interaction solution. On the core functionality side, the system still needs to transform the original medical formats to a common language (functionality accomplished by the *formatMapper* service) and has to be able to process queries and data information as a result of the user interaction (*queryProcessor* service). Finally, there exist now a much independent support for image processing and storage. By specifying both the *imageProcessing* and *DBManager* services we ensure that the system remains independent from the storage policy and from the processing capabilities. The database support now is understood as the access to services in charge of controlling and managing a storage resource and the inner business logic is modelled as specialized services composed following an orchestration coordination strategy.

Once the PIM level view of the architecture has been modelled, several other models should be created as indicated by the SOD-M method and the MIDAS framework. Due to space limitations, the concrete models have been left out of the article. The reader might refer to the bibliography to check how the hypertext of the system [20], the concrete behaviour [7] and the database schema at PIM level [26] should be modelled.

In order to obtain a workable version of the system, the features of the specific implementation technology should be included in the already designed models. This will be done through the specification of the models at the PSM level. The corresponding architectural model of the system at PSM level is shown in Figure 3.

That model for the platform dependent level must represent which technologies are used to implement each of the services that build the software solution. In that sense, this model will represent services, understood as computational entities providing functionalities through a well known interface; service components, identified with components not necessarily connected to web technologies, i.e., components usually found on hybrid architectures such as pre-existent components or COTS (*Commercial Off-The Shelf*) ones; and, finally, the resources that are managed, controlled and accessible through a service.

Figure 3 depicts service components such as *imageViewerInterface* and *uploaderInterface* that will be implemented, in a first step, using a Web interface (as indicated by the value given to the *Tech* property) to maintain the original MEDiWIS functionality. The *formatMapper* service is considered to be a computational entity that will be built upon the standard Web Service technology [27]. The composite service specified in the PIM model is split into two Web services at the PSM level: *resultBuilder* and *infoManager*. The main task of the former will be to prepare the images to be processed and invoke the external *imageProcessing* service. The *infoManager*, in turn, will be responsible of executing, among other responsibilities, the workflow associated to the retrieval, transformation and subsequent storage of results. Apart from the *imageProcessing* service, there exist two more external services: *BankService*, in charge of managing the economical information related to the operations performed by any user over the system; and *DBAccessManager*, which is a Web Service that represents the unique way to access and manage the external storage for the digital medical images and related information.

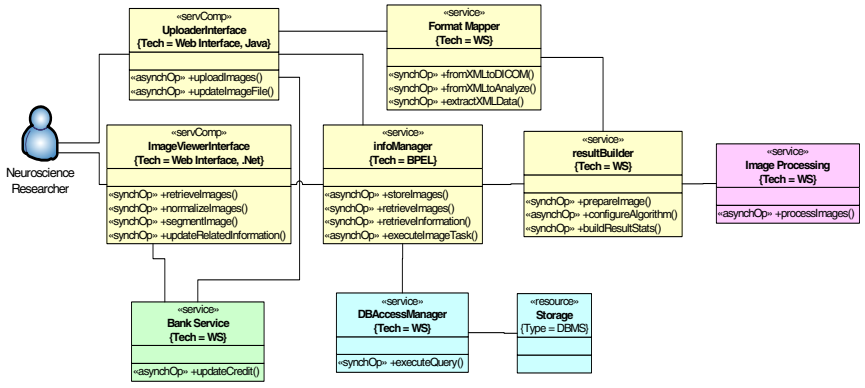

Fig. 3. Service-oriented MEDiWIS: PSM-level architectural model

The *imageProcessing* service is one of the most important services at the PSM level. From the point of view of the capability of the system, as it has been implemented as an external and independent entity, it allows the system to be upgraded and extended with new functionalities very easily. At this moment, this service is only a simple Web service in charge of image normalization and segmentation, however, we are currently moving the algorithms to a Grid Computing [9] environment, and, as a result, this service will be much more complex in a near future.

The contracts that were specified in the PIM-level architectural model have been transformed into simple communication connections among the different services identified at the PSM-level. This is due to the fact that the way in which the communication among two elements is implemented will depend on the technology selected. In most of the cases, the communication control will simply be embedded in the implementation of the elements connected (a Web Service invocation from the Web interface will be coded within the code of the server Web page). In the cases in which a more complex communication takes place, i.e., in *Dialogue* service contracts, a 'mediator' element will be needed. This element will implement the message exchange pattern and control the communication as the involved services shift turns.

4 Related Works

In this section we study the most representative proposals concerning service enactment and definition, software architecture modelling and model-driven service development.

About the identification of the elements to appear in service architectures there are different points of view. Baresi et al.[5], for example, consider 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. Starting from that proposal, Heckel et al. [11] define a UML profile for SOA modelling. Their UML profile only allows to

define two distinct kinds of services (provider and registry) and does not include any facility to model semantic or constraint. Wada et al. [29] 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 only first-class elements.

There are other proposals whose set of concepts is more complete and adequate for the definition of a general SOA model. Papazoglou et al. [23], for example, make a classification of the concepts involved in SOA development but from the viewpoint of the role given to each component. Autili et al. [4], in turn, define a complete service concept model for the development of service-based applications focusing on quality of service features in pervasive systems. This approach is also worth mentioning as it is one of the few proposals that include a specific element to handle user interaction.

Several other proposals have come up from the enterprise research world; however, most of them are based on Web Services technologies. This circumstance appears in works such as the ones by Aiello et al. [2], Lublinsky et al. [18], Erl [8], or Krafzig et al. [14]. This is mainly due to the fact that they start from the definition of service (and SOA by extension) given by the W3C [28], 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 generic level and supporting technologically neutral architectures.

Concerning the definition of the architecture itself, that is, the architectural model containing the service concepts and elements, most of the proposals use UML as notation. They develop service metamodels ([30]) or UML profiles to be used in SOA modelling ([11][29]). Following the principles of Web services, Amir et al. [3] 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. It is also worth mentioning the proposal by IBM and its UML 2.0 profile [13], which we think is one of the most complete and accurate approaches for SOA modelling. There are also proposals of ADLs for the description of SOA such as the ones by Krüger et al. [15] or Rennie et al. [24], but this is a topic beyond the scope of this article.

As MDD is currently a leading trend in software development, many initiatives follow this approach. For example, Zdun et al. [30] use models to integrate process-driven service-oriented architecture models. Heckel et al. [11] define a UML profile for SOA modelling taking into account the MDA principles, that is, the separation of PIM and PSM levels.

In summary, there are not many proposals which gather simultaneously all the features of the work presented: first, all the concepts involved in SOA: the enactment of service composition, the description of service boundaries with the environment, the definition of architectural constraints within the model, etc.; second, a sufficient abstraction level, enough to be independent of the target platform and underlying technologies in order to allow a flexible development of service-based solutions; and, third, a methodological solution in which SOA architectural models play the guiding role throughout the entire software development process.

5 Conclusions

In this work we have presented the redesign of a system for the storage and management of medical digital images called MEDiWIS. The output we have obtained during the redesign process has served us to check the suitability of the SOD-M method for the development of next generation software following an ACMDA approach and service-oriented.

By using an architecture-centric perspective we ensure, first, an adequate control of the system evolution since the architectural model will reflect all the structural and behavioural changes that affect the system; and, second, the proper way of representing any reconfiguration occurred within the system and its elements.

By following a model-driven approach we allow the semi-automatic evolution not only during the software lifecycle (through the control of the architectural changes) but, more importantly, during the software development process itself. In order to achieve automatic transitions form one model to another, transformation rules should be defined. However, and due to space limitations, we have left this aspect for future work.

Finally, by taking into consideration the principles of the MDA proposal, we benefit from a model architecture that allows us to clearly separate and identify the main business domain needs at high level (CIM level). Moreover, by having a model describing the software solution at a platform and technologically 'agnostic' level, we define a satisfactory support for interoperability among systems. Models defined at PIM level also favour the migration of the system from one platform to another. This goal will be achieved just by indicating the target technology of election at PSM level and defining its correspondent transformation rules.

Acknowledgements

This research is partially funded by projects MODEL-CAOS (TIN2008-03582/TIN) and AT (CONSOLIDER CSD2007-0022, INGENIO 2010) from the Spanish Ministry of Science and Innovation.

References

[1] Acuña, C.J., Marcos, E., de Castro, V., Hernández, J.A.: A web information system for medical image management. In: Barreiro, J.M., Martín-Sánchez, F., Maojo, V., Sanz, F. (eds.) ISBMDA 2004. LNCS, vol. 3337, pp. 49–59. Springer, Heidelberg (2004)

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

[3] Amir, R., Zeid, A.: A UML profile for service oriented architectures. In: OOPSLA 2004 Companion, Vancouver, Canada, October 2004, pp. 192–193 (2004)

[4] Autili, M., Cortellessa, V., Di Marco, M., Inverardi, P.: A Conceptual Model for Adaptable Context-aware Services. In: Proc. of WS-MaTe 2006, Palermo, Italy (2006)

[5] Baresi, L., Heckel, R., Thone, S., Varro, D.: Modeling and validation of service-oriented architectures: Application vs. style. In: Proc. ESEC/FSE 2003, Helsinki, Finland (September 2003)

[6] Cáceres, P., Marcos, E., Vela, B.: A MDA-Based Approach for Web Information System Development. In: Workshop in Software Model Engineering (2003)

[7] De Castro, V., Marcos, E., López-Sanz, M.: A model driven method for service composition modelling: a case study. Int. J. Web Eng. Technol. 2(4), 335–353 (2006)
[8] Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, Upper Saddle River (2005)
[9] Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kauffmann, San Francisco (1998)
[10] Greenfield, J., Short, K., Cook, S.: Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. John Wiley & Sons, Chichester (2004)
[11] Heckel, R., Küster, J., Thöne, S., Voigt, H.: Towards a UML Profile for Service-Oriented Architectures. In: Proc. of MDAFA 2003, Enschede (June 2003)
[12] Hernandez, J.A., Acuna, C.J., de Castro, M.V., et al.: Web-PACS for Multicenter Clinical Trials. IEEE Trans. on Inf. Technology in Biomedicine 11(1), 87–93 (2007)
[13] Johnston, S.: UML profile for software services. IBM DeveloperWorks Site (13 April 2005), http://www-128.ibm.com/developerworks/rational/library/05/419_soa/
[14] Krafzig, D., Banke, K., Slama, D.: Enterprise SOA: Service Oriented Architecture Best Practices. Prentice Hall PTR, Upper Saddle River (2004)
[15] Krüger, I.H., Mathew, R.: Systematic Development and Exploration of Service-Oriented Software Architectures. In: Proc. of WICSA 2004, Oslo, Norway, pp. 177–187 (2004)
[16] Kush, R.: Clinical Data Interchange Standards Consortium. In: Proc. of CDISC 2003 (2003), http://www.cdisc.org/pdf/CDISC2003RebeccaKush.pdf
[17] López-Sanz, M., Acuña, C.J., Cuesta, C.E., Marcos, E.: Defining Service-Oriented Software Architecture Models for a MDA-based Development Process at the PIM-level. In: Proc.of WICSA 2008, Vancouver, BC, Canada (2008)
[18] Lublinsky, B.: Defining SOA as an architectural style: Align your business model with technology. IBM DeveloperWorks site, http://www-128.ibm.com/developerworks/webservices/library/ar-soastyle/index.html
[19] Marcos, E., Acuña, C.J., Cuesta, C.E.: Integrating Software Architecture into a MDA Framework. In: Gruhn, V., Oquendo, F. (eds.) EWSA 2006. LNCS, vol. 4344, pp. 127–143. Springer, Heidelberg (2006)
[20] Marcos, E., Cáceres, P., de Castro, V.: Modeling the Navigation with Services. In: ICWI 2004, pp. 723–730 (2004)
[21] OASIS. Reference Model for Service Oriented Architecture. Committee draft 1.0, http://www.oasis-open.org/committees/wd-soa-rm-cd1ED.pdf
[22] Miller, J., Mukerji, J. (eds.): OMG. Model Driven Architecture. Document No. ormsc/2001-07-01, http://www.omg.com/mda
[23] Papazoglou, M.P.: Service-Oriented Computing: Concepts, Characteristics and Directions. In: Proc. of WISE 2003, Roma, Italy, December 10-12, pp. 3–12 (2003)
[24] Rennie, M.W., Misic, V.B.: Towards a Service-Based Architecture Description Language. TR 04/08, Technical Report, University of Manitoba, Canada (August 2004)
[25] Stahl, T., Voelter, M., Czarnecki, K.: Model-Driven Software Development: Technology, Engineering, Management. John Wiley & Sons, Chichester (2006)
[26] Vela, B., Marcos, E.: Extending UML to represent XML Schemas. In: Eder, J., Welzer, T. (eds.) Proc. of CAISE 2003 FORUM (2003)
[27] Vela, B., Acuña, C., Marcos, E.: A Model Driven Approach for XML Database Development. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 780–794. Springer, Heidelberg (2004)
[28] W3C. Web Services Activity and Standards, http://www.w3.org/2002/ws/
[29] Wada, H., Suzuki, J., Oba, K.: Modeling Non-Functional Aspects in Service Oriented Architecture. In: Proc. of the ICSOC 2006, Chicago, IL (September 2006)
[30] Zdun, U., Dustdar, S.: Model-Driven Integration of Process-Driven SOA Models. Intl. J. of Business Process Integration and Management 2(2), 109–119 (2007)