Modeling Adaptable Business Service for Enterprise Collaboration

Khouloud Boukadi, Lucien Vincent and Patrick Burlat

Division for Industrial Engineering and Computer Sciences, ENSM, Saint-Etienne, France
{Boukadi, Vincent, Burlat}@emse.fr

Abstract. Nowadays, a Service Oriented Architecture (SOA) seems to be one of the most promising paradigms for leveraging enterprise information systems. SOA creates opportunities for enterprises to provide value added service tailored for on demand enterprise collaboration. With the emergence and rapid development of Web services technologies, SOA is being paid increasing attention and has become widespread. In spite of the popularity of SOA, a standardized framework for modeling and implementing business services are still in progress. For the purpose of supporting these service-oriented solutions, we adopt a model driven development approach. This paper outlines the Contextual Service Oriented Modeling and Analysis (CSOMA) methodology and presents UML profiles for the PIM level service-oriented architectural modeling, as well as its corresponding meta-models. The proposed PIM (Platform Independent Model) describes the business SOA at a high level of abstraction regardless of techniques involved in the application employment. In addition, all essential service-specific concerns required for delivering quality and context-aware service are covered. Some of the advantages of this approach are that it is generic and thus not closely allied with Web service technology as well as specifically treating the service adaptability during the design stage.

Keywords: service oriented architecture, model driven architecture, adaptable business service, context-aware.

1 Introduction

Service oriented architecture (SOA) plays an ever more important role in modern networked economies. SOA can reinforce business aspects with a flexible infrastructure thanks to independent, reusable automated business processes called services. This trend is a leading paradigm shift in enterprise structure from the traditional single entity to a collaboration of services. So from this perspective, enterprise collaboration can be seen as a conglomeration of outsourced business services on the Web, cooperating to achieve a shared goal. Although the hype surrounding SOA is widespread, the concept is still in its infancy with regards to

actual implementations. Several problems have arisen. Some authors believe that technology and standards are crucial to achieve the aims of SOA but they are not sufficient on their own [1]. Issues like the lack of precise definitions for the SOA concepts involved and the enormous demand for process guidance and proven best practices in these projects are among the most frequently discussed in both industry and academia.

Previously, we discussed in [2] the importance of context-aware business services that fit dynamic enterprise collaboration and demonstrated that business services must have the capacity to adapt their own behavior by configuring appropriately to the situation in which they evolve. Following those conclusions, the main issue of the present endeavor is to propose a conceptual framework. This will consider adaptable business service as a base class modeling entity, to provide appropriate guidance for SOA modeling. We believe that in order for successful service-oriented development, SOA should be described at a high level of abstraction regardless of implementation details. Besides, business services need to be updated or altered in order to follow the rapid shifting trends or demands of e-commerce. These kinds of issues related to reusability, customizability and adaptability of the service development needs to be considered early in the service-system development. One of the current influential trends is Model-Driven Architecture (MDA). The ideas behind MDA can be used to facilitate and improve SOA development [3]. MDA is a framework for software development driven by the Object Management Group (OMG). Within MDA, models are considered as core class elements during system design and implementation. MDA is based on a separation of the development process in three abstraction levels, namely Computational Independent, Platform Independent and Platform Specific Models, respectively CIM, PIM and PSM [4].

The contribution of this paper is twofold. First, a Contextual Service Oriented Modeling and Analysis (CSOMA) is presented and framed into the MDA framework. Second, UML profiles for the PIM-level service-oriented architectural modelling, as well as its corresponding meta-models, are proposed. These UML profiles cover all essential service-specific concerns required for delivering quality and context-aware services.

The remainder of this paper is organized as follows. In Section 2, the Model Driven Architecture (MDA) and the service oriented architecture are outlined. The CSOMA is presented in Section 3. Then, in Section 4, our UML profiles defined to support the representation of adaptable business SOA concepts are presented. Section 5 details some related work for comparison. Finally, a conclusion and possible further research is discussed.

2. Background

2.1 Model Driven Architecture (MDA)

As defined by the Object Management Group, Model-Driven Architecture (MDA) is "a way of writing specifications, based on a platform-independent model. A complete MDA specification consists of a definitive platform-independent base UML model, one or more platform-specific models and interface definition sets, each describing how the base model is implemented on a different middleware platform" [4].

The core paradigm of MDA is model transformation. With MDA, system construction consists of a sequence of models and transformations among these models. The MDA approach separates the business and application logic by presenting the static and dynamic aspects of a system as a high level abstraction with the implementation details hidden. The abstraction is called the *platform-independent model* (PIM). The MDA also supports a *platform-specific model* (PSM), which contains enough implementation information that can convert it to particular source codes.

2.2 Service Oriented Architecture (SOA)

There are a multitude of definitions for SOA. Furthermore, they are not always pointing in the same direction and often discussing the subject at different levels of abstraction. However, many authors agree that SOA is not a product but rather an architectural style or concept. Some pertinent definitions will be presented below.

Thomas Erl defines an SOA as: "an **open**, **agile extensible**, **federated**, **composable architecture** comprised of autonomous, capable, vendor diverse, interoperable, discoverable and potentially reusable services" [5]. A business definition of the SOA is presented in [6]: "SOA is a conceptual business architecture where business functionality is made available to SOA users as shared, reusable services on an IT network. Services in an SOA are modules of business or application functionality with exposed interfaces, and are invoked by messages". The W3C minimally defines the SOA as "a set of components which can be invoked, and whose interface descriptions can be published and discovered" [7].

The three definitions presented above are on different granular levels. However, they are all cohesive. The definition of [5] and [6] has a wider perspective of SOA then [7], focusing more on architectural features. While the W3C provides a very technical definition that considers what can be done with a service but does not refer to the architecture nor give any additional information on how a service should be designed or configured.

3. MDA and Contextual Service Oriented Modeling and Analysis

The research work presented in this paper is part of our methodology: Contextual Service Oriented Modeling and Analysis (CSOMA) [8]. CSOMA is vendor-independent and provides a systematic approach and a well defined process to guide the design and development of an SOA. CSOMA is the result of applying two kinds of SOA practices: IT SOA and Adaptable Business SOA. IT SOA is based on standard methods to identify IT services. The intent of Business SOA is to offer services which will support business goals of the enterprise. CSOMA delivers a set of business services that suit enterprise collaboration.

One of the most important characteristics of CSOMA is its support of the MDA. As can be seen in Fig.1, CSOMA is organized into three layers aligned with the MDA layers. Each layer has specific architectural characteristics and addresses some particular concerns, as described below:

The CIM level for CSOMA: describes the business logic defined within enterprise's business models and business processes. These models constitute the basis for the business services identification and modeling. Examples of the CIM level include business motivation models, business process models and enterprise architecture, etc.

The PIM level for CSOMA: includes the PIM for Adaptable Business SOA and describes the business service models in a technology independent manner. These models are created by refining the business model identified in the CIM level. Each model defines new modeling elements which extend the UML meta-model [9]. The process describing the tasks associated to the generation of each business service model as well as the mapping rules among them is out of the scope of this paper.

The identified business services models are completed by designing and analyzing service variability among various clients and diverse service contexts. The aim is to design business services that are more than a functionality provided through the Web. Indeed, they must have the capacity to adapt their behavior by providing the appropriate function or capacity which accommodates the current situation. Recently, the aspect-oriented programming (AOP) has gained growing acceptance. AOP furnishes an abstraction and encapsulation mechanism with the purpose of enhancing separation of concerns [10]. This aspect based separation is applied to the service modeling process. In the PIM for Adaptable Business SOA, an aspect-oriented technique is used to help identify and model the crosscutting concerns and separate them from the services functions during the service modeling process. Hereafter, these aspects will be referred to as "Conceptual Aspects".

The PSM level for CSOMA: includes models relating to the Web services technology. The business services models should be mapped and applied over the Web service technology. Example of these models could be BPEL, Web services Models etc. Note: this topic is beyond the scope of this paper.

The focus in this paper is in the proposition of UML profiles for the PIM-level service-oriented architectural modeling.

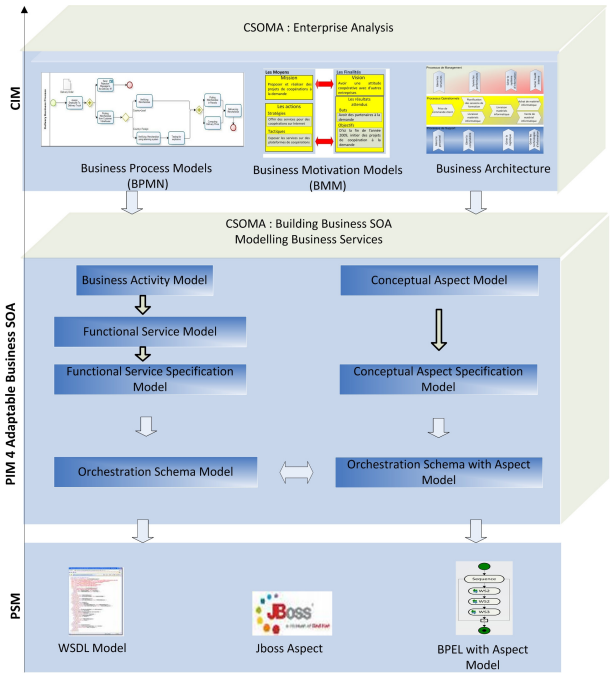

Fig. 1. CSOMA and MDA

4. UML Profiles for Adaptable Business Service Oriented Architecture

In this section we introduce our aforementioned UML profiles for their use in the definition of the adaptable business SOA at PIM-level. A UML profile extends the standard UML elements in order to precisely describe specific domain or application concepts [11]. Our Adaptable Business SOA UML profiles consist of two profiles:
- *Business service profile*: describes how to model a service including its description and interactions.
- *Conceptual Aspect profile*: extends the business service profile and describes how to model context-aware business services.

Next the concepts in the UML profiles using the associated meta-models will be described.

4.1 Concepts involved in the Business service profile

Fig. 2 presents the business SOA meta-model using UML notations. When dealing with a business SOA, there are four different concepts for service: the business idea,

as well as the service's description, interactions and adaptability. These four concepts are explained.

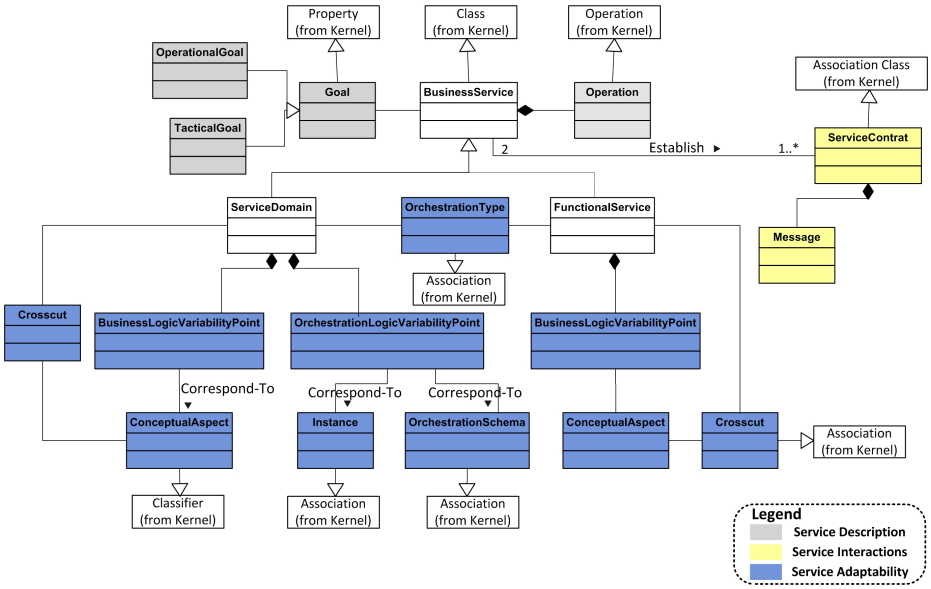

Fig. 1. Adaptable Business SOA meta-model

Business Service: is treated as core class modeling entity in our business SOA meta-model. The business services inside our CSOMA methodology play different roles. They can be classified according to their atomicity and scope. The business services can be atomic services called functional or composite services called service domains. Functional services are fine grained services exposing a set of business activities. Examples of functional services can be *computing price for delivery* or *delivering merchandise*. Service Domain orchestrates a set of functional services in order to provide high level of functioning as well as a comprehensible external view to the end user. Service domain will be published as a composite service, thus reducing the complexity of publishing, selecting and combining fine grained functional services. This composite structure is used as major building block for implementing on demand enterprise collaboration.

Service Description: business services have a set of operations (serviceOperations) and fulfill goals (Goal). We define service operations as atomic functions that are used to accomplish the service goal. A goal is 'an optative' statement, an intention that expresses what is wanted [12]. Goal can be expressed at different levels of abstraction starting from tactical to operational. Functional services relate to operational goals, i.e. goals that are achieved with tasks provided by the service. Services domains correspond to high level aims (tactical). For example, *take customer order* service domain fulfils an enterprise tactical goal which is "Use the Web to expose enterprise products". While a computing delivery price, which is a functional service, achieves a "Managing Customer Invoice" goal. This is an

operational goal which can be fulfilled by the service operations. These two elements (operations and goals) constitute the service description at the PIM level.

Service Interactions: business services communicate and interact with each other through contracts. Within the SOA, service contract establishes the terms of engagement and interaction among the different architecture components. These interactions include service contract and messages flowing between services.

- Service Contracts: contracts established between services must reflect the services that are involved in the contract, the roles played, and several other properties such as, purpose or contract expiration time. Other authors like [13] treat the service contract as part of the service description itself. Even though the way to interact with a service depends generally on its operations shown in its interface, we propose that, by making the service contract an independent element of the model, the independence of the different relations that a service can establish with other services, each with its own restrictions and characteristics is assured.

- Messages: messages in SOA represent the communication exchanged between services. Each message is related to a service contract and has meaning to both the service and the consumer. Some authors suggest that the message format and its addresses such as HTTP or SMTP should be included in the model [14]. We consider that the format of a message is an issue that depends mostly on the implementation technology and so it should be modeled in the correspondent PSM-level models.

Business service adaptability: business services must have the capacity to alter their own behavior to accommodate the situations as they occur. To meet to this objective, service designers should model services with points that mark where the service can be customized depending on the customer and the environment context. We call these, variability points, similar to those variability points that are introduced in software product line engineering [15]. Depending on the business service itself (service domain or functional service), we identify two types of variability points. Functional service has a set of variability points which concern essentially its business logic. While a service domain has variability points concerning its business logic as well as its orchestration logic. Our hypothesis is that more the modeling of the business service is parameterized, more its adaptation is facilitated. At the design phase, the variability points, concerning the orchestration logic, deal with the creation of a set of orchestration schemas for a single service domain. When the service domain receives an incoming request, it selects a suitable schema. The selection of the orchestration schema takes into account the context of the incoming request. The variability of orchestration logic was the object of our research project in [16], so further details are provided there. In our present work, the variability points concerning the business logic, which are handled through Conceptual Aspects, are examined. These are explained in the next section.

4.2 Concepts involved in the Conceptual Aspect profile

We define Conceptual Aspect as a domain-specific concern for a business service which groups non functional requirements or designates a business rule or addresses an adaptability action in response to contextual information. According to our definition, three types of Conceptual Aspects can be distinguished:

The *Non-Functional Aspect* is used to classify the quality attributes for a service such as security, performance, availability and so on. Its objective is to provide a behavioral guideline on service or its element, as well as means to control one or several services. For example, a non-functional aspect for security enforces the control of access to services.

The *Business Rule Aspect* defines or constrains some concerns of the business. It is intended to control the behavior of the business. As its name suggests, this aspect encapsulates a rule that can be: constraint, action enabler, and computation or inference rules.

The *Context Enabler Aspect* is used to personalize the basic functions of services according to customer's context or preference. The Context Enabler aspects try to satisfy customer preferences and to improve convenience by altering the messages or operations in a service. For example, in a Delivery Service Domain, we can define an Aspect related to the calculation of extra fees when there is a context change that corresponds to a modification in the delivery date.

The Conceptual aspects need to be explicitly and coherently modeled, their design should not be scattered throughout the business service model. We propose to use Aspect Oriented Programming principles to model Conceptual Aspects. AOP introduces unit of modularity called Aspects containing different code fragments (*advice*), and location descriptions (*pointcuts*) to identify where to plug the code fragment. The points which can be selected by the *pointcuts* are called *join points* [17]. Fig.3 presents the meta-model of a Conceptual Aspect, which extends the one presented in Fig. 1. The advice is represented by the advice stereotype that can be applied to a service operation. Advice is linked to a business service by the <<crosscut >> association.

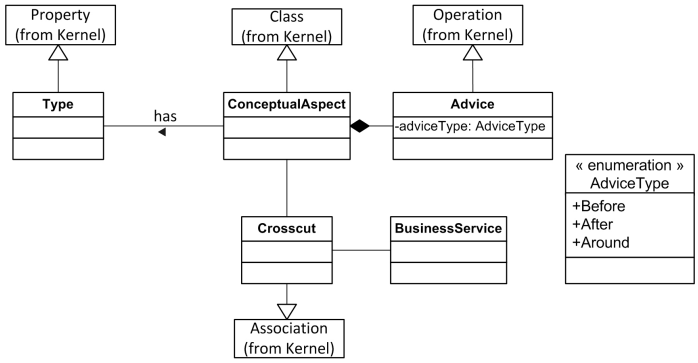

Fig. 2. Conceptual Aspect meta-model

5. Related Work

This section reviews some publications in literature in order to better illustrate the position as well as the novel aspects of our endeavors. There are many ongoing

research efforts related to the model-driven development of SOA systems such as [18], [19]. Baresi et al. propose static SOA model and dynamic model in UML [20]. Nevertheless, these authors define the models into a transition system and specify a set of transformation rules to support the dynamic reconfiguration in SOA. This work differs in that it assumes that any SOA development is built upon service clients, providers and service discovering agents. Otherwise, the scope in which SOA can be used is constrained since it cannot be generalized to other execution platforms apart from the ones that follow their predesignated schemas.

Heckel et al. already proposed an excellent UML profile for SOA modeling that takes into account the MDA principles [18]. However, their UML profile defines only two kinds of services (provider and registry) and does not include any facility to model adaptive services.

MIDAS is a first rate model-driven methodology for Web information system development based on MDA principals [21]. The methodology does suggest some PIMs, PSMs and mapping rules between the models. The methodology uses the UML to represent the different models. At the PIM level, they propose a UML profile for SOA. Their work is one of the closest publications to the content presented in this article. However, the authors do not consider the service adaptability. To our knowledge, our paper is the first to adopt service adaptability at the PIM level.

IBM presents a UML 2.0 profile for software services [22]. The profile includes messages, specification of a service, manner in which services are composed into aggregate services, choreography view of services, and policy perspective. This similar research presents SOA concepts not so constrained to the Web service technology. However, once again this does not deal with our focus on the business service adaptability at the PIM level of the MDA.

6. Conclusion

Service oriented architecture presents a promising integration approach to enable inter-enterprise collaboration and deliver maximum reusability, agility and flexibility when facing changing conditions. However, its implementation in actual enterprise contexts is more challenging and requires particular attention, comprehensive guiding framework and strong design principles. In this paper, we highlight the importance of the application of the MDA principles to the SOA paradigm. We presented UML profiles for the design of PIM-level Business SOA. These profiles covers all concerns required for delivering high-quality and context-aware services. The modeled business services constitute the building blocks for on demand enterprise collaboration. As to future work, of course, an empirical study to validate and test the proposed approach will be at the centre of ongoing research. For other related endeavors, integrating or taking into account the behavior of business services and defining an appropriate specification for them appear to be a promising path as well.

References

1. Erradi, A., S. Anand, and N. Kulkarni. SOAF: An Architectural Framework for Service Definition and Realization. in IEEE International Conference on Services Computing (SCC'06), pp. 151 -158. 2006.
2. Boukadi, K., C. Ghedira, and L. Vincent. An Aspect Oriented Approach for Context-Aware Service Domain Adapted to E-Business. in The 20th International Conference on Advanced Information Systems Engineering. 2008. Montpellier-France
3. Lopez-Sanz, M., C.E. Cuesta, and E. Marcos, Modelling of Service-Oriented Architectures with UML. Electronic Notes in Theoretical Computer Science, 2008. 194(4): p. 23-37.
4. OMG, MDA guide version 1.0.1, proposed by the Object Management Group. 2003.
5. Erl, T., Service-Oriented Architecture (SOA): Concepts, Technology, and Design 2005: Prentice Hall 792.
6. Marks, E.A. and M. Bell, Service-Oriented Architecture : A Planning and Implementation Guide for Business and Technology. 2006: First Edition,New Jersey: Wiley, 2006.
7. W3C, Web Services Glossary, available at: http://www.w3.org/TR/ws-gloss. 2004.
8. Boukadi, K., L. Vincent, and P. Burlat, The Contextual Service Oriented Methodology (CSOMA), research report available at: http://www.emse.fr/~boukadi/. 2009.
9. UML, UML Superstructure 2.0, OMG Adopted Specification PTC/03-08-02, available at: http://www.uml.org/. 2003.
10. AOP, Aspect-Oriented Software Development, available at: http://www.aosd.net. 2007.
11. OMG, Object Management Group, UML2.0 Super Structure Specification, Octobre 2004. 2004.
12. Rolland, C. and R.S. Kaabi. An Intentional Perspective to Service Modeling and Discovery. in Proceedings of 31st Annual International Computer Software and Applications Conference, COMPSAC 2007. 2007: IEEE Computer Society
13. Krafzig, D., K. Banke, and D. Slama, Enterprise SOA Service Oriented Architecture Best Practices. 2004: Upper Saddle River: Prentice Hall PTR.
14. Amir, R. and A. Zeid. An UML Profile for Service Oriented Architectures. in Companion to the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2004. 2004.
15. Jaring, M. and J. Bosch, Variability dependencies in product family engineering, in Software Product-Family Engineering S.B. Heidelberg, Editor. 2004. p. 81-97.
16. Boukadi, K., et al. CWSC4EC:How to Employ Context, Web Service, and Community in Enterprise Collaboration. in In the 8th International Conference on New Technologies of Distributed Systems (NOTERE 2008). 2008. Lyon, France.
17. Kiczales, G., et al. Aspect-Oriented Programming. in The European Conference on Object-Oriented Programming (ECOOP). 1997. Finland.
18. Heckel, R., et al. Towards a UML Profile for Service-Oriented Architectures. in Workshop on Model Driven Architecture: Foundations and Applications (MDAFA '03), University of Twente, Enschede, June 2003. 2003.
19. Zhang, X.G. Model Driven Data Service Development in IEEE International Conference on Networking, Sensing and Control, ICNSC 2008. 2008. China.
20. Baresi, L., et al. Modeling and validation of service-oriented architectures: Application vs. style. in the 9th European Software Engineering Conference (ESEC/FSE 2003). 2003. Helsinki, Finland, September 1-5.
21. Cáceres, P., E. Marcos, and B. Vela. MDA-based approach for web information system development. in Proceedings of Workshop in Software Model Engineering. 2003.
22. Johnston, S., UML profile for software services, in IBM DeveloperWorks, April 2005, URL:http://www-128.ibm.com/developerworks/rational/library/05/419_soa. 2005.