Towards Adaptable SOA: Model Driven Development, Context and Aspect

Valérie Monfort[1,2] and Slimane Hammoudi[3]

[1] Université Paris IX Dauphine LAMSADE
Place du Maréchal de Lattre Tassigny, Paris Cedex 16 France
[2] Université Paris 1 - Panthéon –Sorbonne
Centre de Recherche en Informatique, France
[3] ESEO 4, Rue Merlet de la Boulaye B.P. 926
49 009 ANGERS Cedex 01 France
valerie.monfort@univ-paris1.fr, slimane.hammoudi@eseo.fr

Abstract. Service-Oriented Architectures (SOA) are broadly used by companies to gain in flexibility. Web service is the fitted technical solution used to support SOA by providing interoperability and loose coupling. However, there is still much to be done in order to obtain a genuine flawless Web service, and current market implementations still do not provide adaptable Web service behavior depending on the service contract. In this paper, we propose two different approaches to increase adaptability of Web services and SOA. The first one is based on a technical solution which considers Aspect Oriented Programming (AOP) as a new design solution for Web services. We implemented an infrastructure to enrich services with aspects and to dynamically reroute messages according to changes, without redeployment. The second one combines Model Driven Development (MDD) and Context-Awareness to promote reusability and adaptability of Web services behavior depending on the service context.

Keywords: Aspect Based services, Meta Modeling, Model Composition.

1 Introduction

Economical context impacts companies and their Information System (IS). With SOA (Service Oriented Architecture), each application owns interfaces, offering services and masking implementation details. Applications are seen as black boxes independently connected to a middleware as Enterprise Application Integration bus (EAI) with its connectors (connecting the bus to the applications). However, this integration solution does not allow connecting heterogeneous applications or infrastructures, as distant IS. Web services [4][10] are the cheapest and simplest technical solution to resolve this problem. They offer interoperability because they are based on standards as XML (eXtensible Markup Language) and allow loose coupling. Web services (WS), like any other middleware technologies, aim to provide mechanisms to bridge heterogeneous platforms, allowing data to flow across various programs. The WS technology looks very similar to what most middleware technologies looks like. Consequently, each WS possesses an Interface Definition Language, namely WSDL

L. Baresi, C.-H. Chi, and J. Suzuki (Eds.): ICSOC-ServiceWave 2009, LNCS 5900, pp. 175–189, 2009.

(Web Service Description Language), which is responsible for the message payload, itself described with the equally famous protocol SOAP (Simple Object Access Protocol), while data structures are explained by XML. Very often, WS are stored in UDDI (Universal Description Discovery and Integration) registry. In fact, the winning card of this technology is not its mechanism but rather the standards upon which it is built. Indeed, each of these standards is not only open to everyone but, since all of them are based on XML, it is pretty easy to implement these standards for most platforms and languages. For this reason, WS are highly interoperable and do not rely on the underlying platform they are built on, unlike many ORPC (object remote procedure call). According to a vast majority of industrial leaders, WS is the best fitted technology for implementing Service Oriented Architectures. With WSs, the message contract (WSDL) is the central meeting point which connects applications. The WSDL contract constitutes the design view upon which developers can generate both client and server sides (proxy and stub).

We noticed code is very monolithic it encapsulates different concerns as business, security… Moreover, we used to change Web service code according to new needs, to redeploy Web service. Each change is time costly and Web service is not available. So, flexibility to changes is not optimal. We proposed aspects based solutions to gain in code simplicity without re deploying code with a non intrusive manner [3][22]. We based our more recent approach on extended BPEL (Business Process Execution Language) [2][18] and temporized automatons [1][15], that we prototyped by providing client, and server adaptability.
Nevertheless, we are convinced this pragmatic and efficient solution is too complex for non expert users and developers and difficult to maintain, because it requires strong technical knowledge.

Recently, we have investigated a model driven approach and context awareness to provide developers mechanisms that allow them representing an application in abstract way (in a model) and, then, automatically generating the corresponding code. We broadly discussed about adaptability according to context in [23].

We aimed to explore adaptability and flexibility on a service platform using context with the benefits of an MDD (Model Driven Development) [26] development strategy. These benefits are related to productivity, quality, adaptability and maintenance. Model Driven Architecture (MDA) is based on standards from the Object Management Group (OMG); it proposes an architecture with four layers [24]: meta meta model, meta model, model and information (i.e. an implementation of its model). Object-oriented and component technology seem insufficient to provide satisfactory solutions to support the development and maintenance of these systems. To adapt to this new context, software engineering has applied an old paradigm, i.e. models, but with a new approach, i.e. Model Driven Development (MDD). In this new global trend, MDA is a particular variant. Adaptable Service platforms have been proposed for the development of mobile context-aware applications.

The development of such platforms involves a number of challenges from which we consider two main issues in the context of our approach of model driven development : i) the definition of a meta model to describe the contextual domain in which a given application or service is defined, ii) A mechanism to integrate the context into the business application using a model driven approach.

We propose to discuss about the pertinence to merge these two solutions. The section 2 explains our technical approach based on services and aspects to implement adaptability in Web services. The section 3 shows how our first step researches about MDD approach assure service adaptability while using context mechanisms. The section 4 comments the needs of merging and extending these two research works: aspect based services and context modeling with parameterized transformation. It launches our future research in this topic. Finally, we will discuss our solution and conclude this work. Let us see now Aspect based services.

2 Aspects for Web Services Adaptability

2.1 Applying AOP to Web Services with ASW

Aspect Oriented Programming (AOP) is viewed as an answer to improve Web services flexibility. AOP [5][20] is a paradigm that enables the modularization of crosscutting concerns into single units called aspects, which are modular units of crosscutting implementation. Aspect-oriented languages are implemented over a set of definitions:

1. Joinpoints: They denote the locations in the program that are affected by a particular crosscutting concern.
2. Pointcuts: They specify a collection of conditional joinpoints.
3. Advices: They are codes that are executed before, after or around a joinpoint.

In AOP, a tool named weaver takes the code specified in a traditional (base) programming language, and the additional code specified in an aspect language, and merges the two together in order to generate the final behavior. The weaving can occur at compile time (modifying the compiler), load time (modifying the class loader) or runtime (modifying the interpreter).

We developed an AOP based tool named Aspect Service Weaver (ASW) [3][18][22]. The ASW intercepts the SOAP messages between a client and an elementary Web service, then verifies during the interaction if there is a new behavior introduced (advice services). We use the AOP weaving time to add the new behavior (before, around or after an activity execution). The advice services are elementary Web services whose references are registered in a file called “aspect services file descriptor”. The pointcut language is based on XPath [6]. XPath queries are applied on the service description (WSDL) to select the set of methods on which the advice services are inserted. We extended this approach to BPEL processes. The ASW controls the BPEL process execution instead of intercepting SOAP messages. It is integrated in the BPEL engine in order to interpret the BPEL process and apply the aspect services. It verifies before the execution of each BPEL activity if some Aspect service has to be inserted. Then, it executes the corresponding advice service. We also add a new functionality to the ASW. The tool dynamically generates messages called execute messages, encapsulating the identifier and the interaction protocol of the advice service. These messages are sent to the client to advertise it about a new behavior inserted at runtime. This message is necessary since the new behavior can require new information exchange involving messages not expected by the client, leading to execution failures. We defined a new process algebra

semantics that associates a timed automaton [1] with an abstract process as shown in [13] . Let us see now related works.

2.2 Adaptability with Aspects Based Services : Related Works

In [3], the authors define specific AOP languages to add dynamically new behaviors to BPEL processes. But, neither of these approaches addresses the client interaction issue. The client has no mean to handle the interactions that can be added or modified during the process execution. The Web Service Management Layer (WSML) [22] is an AOP-based platform for Web services that allows a more loosely coupling between the client and the server sides. WSML handles the dynamic integration of new Web services in client applications to solve client execution problems. WSML dynamically discover Web services based on matching criteria such as: method signature, interaction protocol or quality of service (QOS) matching. In a complementary way, our work proposes to adapt a client to a modified Web services. Some proposals have emerged recently to abstractly describe Web services, most of them [7][13] are grounded on transition system models (Labeled Transition Systems, Petri nets, etc.). [9] Introduces WComp middleware model, which federates three main paradigms: event-based Web services, a lightweight component-based approach to design dynamic composite services, and an adaptation approach using the original concept called Aspect of Assembly.

These paradigms lead to two ways to dynamically design ubiquitous computing applications. The first implements a classical component-based compositional approach to design higher-level composite Web Services and then allow incrementing a graph of cooperating services for the applications. This approach is well suited to design the applications in a known, common, and usual context. The second way uses a compositional approach for adaptation using Aspect of Assembly, particularly well-suited to tune a set of composite services in reaction to a particular variation of the context or changing preferences of the users. In these approaches adaptability is resolved with AOP and /or with specific middleware.

We noticed some research works propose to formally specify composite Web services and handle the verification and the automatic composition issues. But, neither of these works proposes to formalize the dynamics of SOA architectures and to handle runtime interaction changes. Even if this solution addresses our (contextual) adaptability and interoperability aims, nevertheless, it may be felt as complex by non expert users or developers.

We are convinced adding an abstraction layer with metamodeling will facilitate usage of this technology and guaranty interoperability. We are also convinced parameterized Meta modeling is the fitted solution to our contextual adaptability need.

3 Context for Service Adaptability with Model Driven Approach

3.1 MDD and Context for Service Adaptability

3.1.1 A Context Meta Model with Example

Previous research works allowed us to define adaptability and context [23]. In the MDD approach, the use of a metamodel not only guarantees a strong and focused

semantics tied to a particular application domain, but also offers a precise abstract syntax and a common representation to any developed model. We are interested in our research in user centered mobile application [7][8]. Thus, we consider that the defining context here is a set of information structured in three dimensions [26] :

- Actor: A person which is a central entity in our system.
- Environment: in which the person evolves and
- Computational entities which are used by a person to invoke services and capture the different states of the environment.

All the information related to the three dimensions can also be shared by other mobile applications. Figure 1 shows our context metamodel. Our metamodel identifies and adds the most relevant and generic contextual entities that will be held in account in modeling any mobile and context aware application. This context metamodel consists of six generic contextual entities and four deduced entities specific to a category of mobile applications. The class "ContextView" groups all contextual entities involved in a given application. It is identified by name attribute and has two types of relation: the aggregation "involves" and the association "belongsTo". The first relation expresses that a given "ContextView" is composed of many "ContextEntity" that are involved in a context-aware application. The second relation "belongsTo" expresses the use of historical context information. A given context entity may have participated in different context views. This information can be helpful in the design of future context views. The second generic entity of the metamodel is the "ContextEntity". As we see on the figure bellow, it is specialized in three generic entities: Actor, ComputationalEntity and Environnement. Actor may be a person or another object that has a state and profile. It evolves in an environment and uses computational devices to invoke services. With the ComputationalEntity, the computational device is used by the actor to access the services and to capture contextual information from the environment. Usually, a mobile device is used in context aware mobile applications, and can obtain information concerning the type of device it is (PDA, laptop, cellular phone…), the application, the network, etc. The environment is constituted of all the information surrounding the actor and its computational device that can be relevant for the application. It includes different categories of information as :(i) Spatial context information can be location, city, building, (ii) Temporal context information comprises time, date, season, (iii) Climate can be temperature, type of weather…. The last entity is a profile. We are convinced this entity is important in any user centred context aware application. In fact, profile is strongly attached to the actor and contains the information that describes it. An actor can have a dynamic and/or a static profile. The static profile gathers information relevant for any mobile context-aware application. It can be the "date of birth", "name" or "sex". On the opposite, dynamic profile includes customized information depending on the specific type of application and/or the actor. It can be goals, preferences, intentions, desires, constraints, etc. For example the goal of a tourist searching for a restaurant is to have dinner. A profile in this case can give information concerning culinary habitude or constraints of a tourist. Let us see now the benefits of parameterized transformation for context binding.

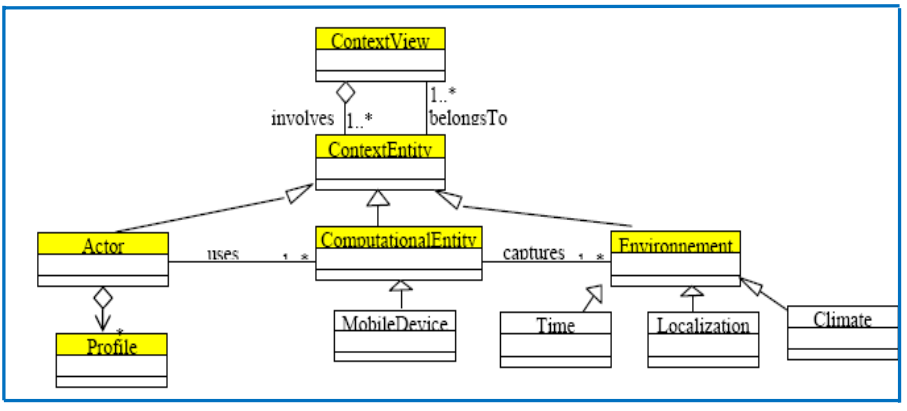

Fig. 1. A context Meta Model

3.1.2 Parameterized Transformation for Context Binding

MOF (Meta Object Facility) is a standard from OMG for meta models specification. The development is based on the separation of concerns (e.g. business and technical concerns), which are afterwards transformed between them. So, business concerns are represented using Platform-Independent Model (PIM), and technical concerns are represented using Platform- Specific Model (PSM). PIM models are more stable over time while PSM models are subject to frequent modification. So, this approach preserves the business's logic (i.e. PIM models) against the changes or evolutions of technologies (i.e. PSM models). The separation of concerns (business and context) is emphasized at a model level of our approach where PIM and context models are defined independently, and then merged by suitable transformation technique. Parameterized transformation allows merging context information with business logic at model level. We have investigated [12] this type of transformation which is not explored and there is not a standard transformation language implementing it. We will discuss shortly this type of transformation. A CPIM model (Contextual Platform Independent Model) is then obtained and fits together business requirements with contextual data. According to [25], "A parameter specifies how arguments are passed into or out of an invocation of a behavioral feature like an operation. The type and multiplicity of a parameter restrict what values can be passed, how many, and whether the values are ordered". In [12] David Frankel mentions the importance of parameterization in model operations using the association of tagged values with PIM and PSM models. Tagging model elements allows the model language to easily filter out some specific elements.

In our proposition these parameters are context or context-aware and after the transformation the application will join the context information specified into the parameters as illustrated in Figure 2. A PIM model can be developed without contextual details. User name, profiles, device type, location can be added as parameters in transformations. The same PIM can be re-transformed and refined many times adding, deleting or updating context information. The designer has to specify into the application model the elements that will receive the context information. A mark, identified by the symbol #, is given for these elements to be recognized by the transformation

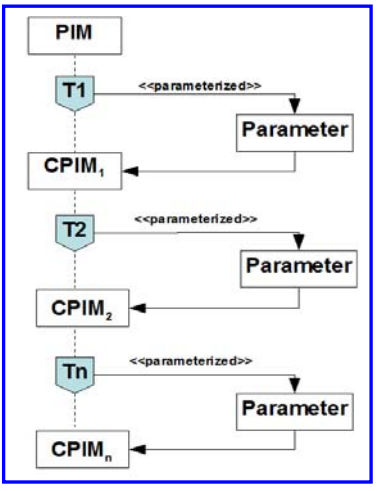

Fig. 2. Parameterized Transformation

engine. The marked elements represent context-aware elements, in others words, the model elements that can be contextualized.

The transformation language must support parameterization techniques. In our case the parameters can be a Context Property and/or a Context Data Type. We use templates to specify which elements in application model are potentially context-aware as depicted in Figure. 3.

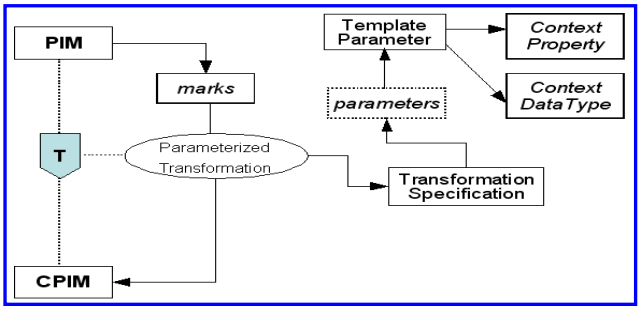

Fig. 3. Parameterized Transformation for context binding

The transformation engine has to navigate into the PIM model verifying the parameters and the elements marked and then make the transformation which consists in an update of contextual properties in a PIM. Template parameter [12] is an element used to specify how classifiers, packages and operations can be parameterized. UML 2.0 presents that any model element can be templateable. For independent context-aware models we need to identify context elements that could be parameterable. A parameterable element is an element that can be exposed as a formal template

parameter for a template, or specified as an actual parameter in a binding of template. Context parameter can be expressed as constraint and compared with the elements signature in template parameter. This operation is named "matching operation". UML presents a Template Signature element that defines the signature for binding the template. Lets us see now related works concerning this approach.

3.2 Adaptability with Context Based Services: Related Works

In [7], the authors propose an UML based context metamodel for the development of context-aware mobile applications implemented on Web services platform. The proposed metamodel does not refine contextual information and focuses on the association between basic contextual structures with service invocation interfaces for both contextual providers and context-aware applications. In [8], they have been applied MDA in context-aware application development. They focus on the development of context-awareness based on ontologies. However, nor context metamodel is proposed neither transformation techniques are used. In [5] authors investigate a number of context models described in the literature and propose a context metamodel based on the main concepts and strengths found on these models. The metamodel is formally described using MOF and has been used as a basis for the development of context-aware applications and an associated service platform. All these works aim to explore adaptability and flexibility on a service platform using context and models. But, neither of these works proposes an explicit approach to integrate context into business logic. By the use of a parameterized transformation technique the contextual parameter identified into the business logic model will be completed and contextualized with the "parameterable elements" which represents context information.

While our approach allows binding contextual data at model level, it doesn't take into account service adaptability which deals with the execution level. In the following section, we discuss this issue of service adaptability using aspect and context in a model driven approach.

4 Towards Models, Context and Aspects for Service Adaptability

4.1 Global Approach

This section aims to present the two techniques of context and aspect could be combined to achieve service adaptability using a model driven approach. Through Model Driven Development, context models are built as independent pieces of application and at different abstraction levels then attached by suitable transformation techniques called parameterized transformation. Context model specify contextual entities that are involved in a given context aware application. From a context model, an aspect model is derived. This aspect model specifies the behaviors linked to the context model. Figure 4 illustrates the main models and transformations techniques involved in our MDD approach. Five main objectives are illustrated:

- A separation between Context Model information (CM) and business logic (PIM) in individual models,

- The derivation of an Aspect Model (AM) from a context model. A Context Model specify the contextual entities with their properties (static view), while the aspect model specify behaviors (dynamic view).
- The integration of the Context Model into the business logic using parameterized transformation techniques. At this stage, the CPIM model is enriched by contextual data but the behavior part for adaptability at execution level is missing.
- The Weaving process add adaptability mechanism producing a CPSM model (Context Platform Specific Model) .
- Finally, a CPSM model is mapped into a service platform for future execution of context-aware services.

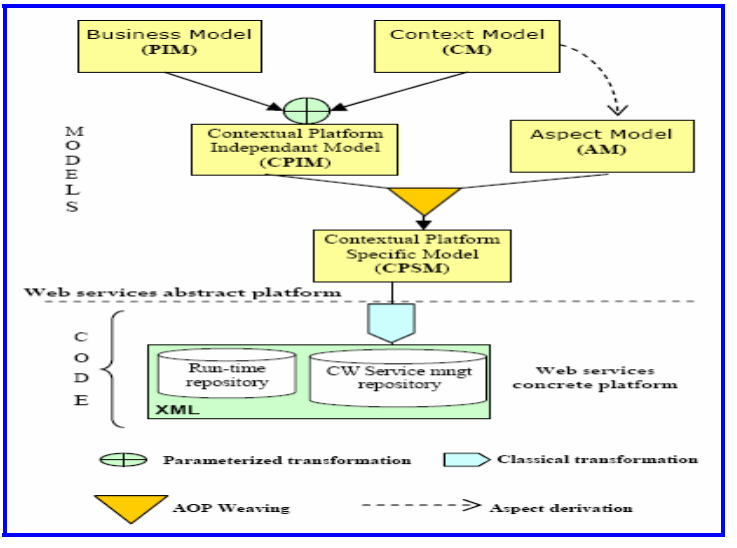

Fig. 4. Models, context and aspect for service adaptability

4.2 Concrete Examples

Figure 5 shows a simplified Business Model (BM) that is underlined. It is comes from a genuine industrial feedback. The company uses to let apartments for holidays (sea side, country side, mountain). If the client chooses an apartment a contract is established according to client profile and apartment characteristics. According to client profile, apartment availability, date in the location schedule, a specific offer may be proposed to the client. So, price will dynamically change according to these parameters. Moreover, Location Manager component will expose an interface with public methods as "ToSelectAppartment" and "ToContract". These methods are services and may be invoked through the Web as Web services in future implementations. Figure 6 expresses a part of an Aspect Model (AM). An aspect includes advices and pointcuts that include pointcut expressions. JoinPoints denote the locations in the program that are affected by a particular crosscutting concern. Advices implements Crosscutting concerns.

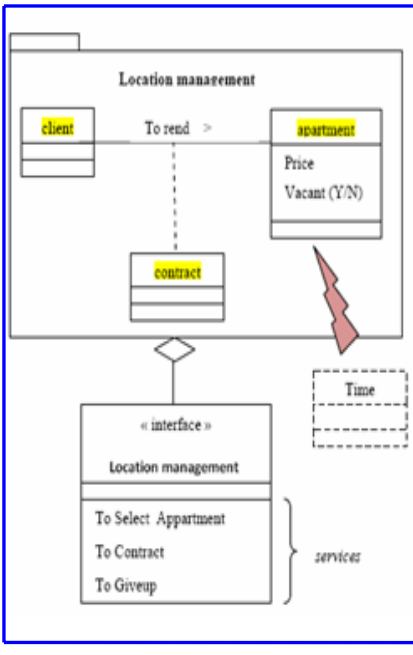

Fig. 5. Business Model including Context

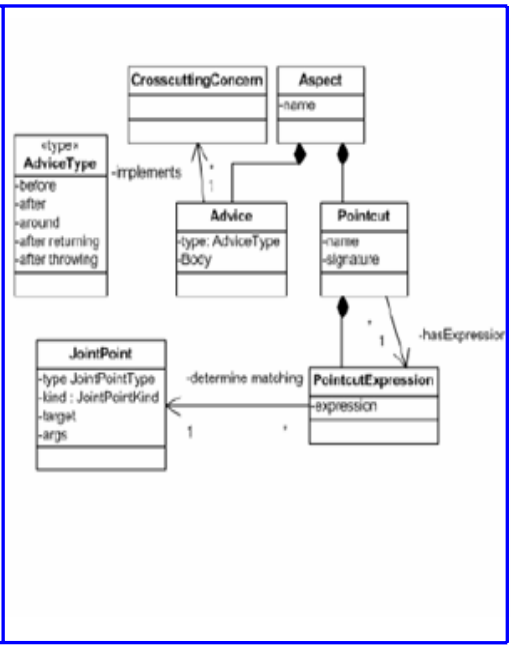

Fig. 6. Partial view of an Aspect Model

4.3 Dynamic Rerouting Modeling (CPIM)

Services are modeled to be orchestrated according to BPMN (Business Process Modeling Notation). On figure 7, we notice "To select apartment" service invokes "To contract" service, but according to previous research works [22] we have to extend BPMN semantics to introduce aspect paradigm and to express:

- Contextual parameters and variability as with vacancy and season parameters
- Message rerouting according to parameters evaluation and context. Here, invocation is dynamically rerouted via "To give up" service.

CPSM includes paradigms of the chosen platform. We chose to extend and ESB (Enterprise Service Bus), an open source called MULE. This ESB will be extended with ASW techniques. We have to develop this modeling and mapping rules still remain to do.

4.4 Model Transformation

Previous research works [26] proposed a static solution by extending OCL (Object Constraint Language) language to adapt some model transformation operations used to attach context into application models. Differently from traditional model transformations, the parameterized one has as source model a set of contextual parameters and as target model the PIM marked model. The designer determines which model

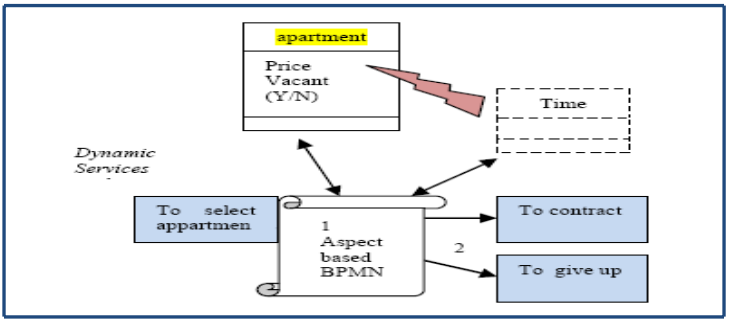

Fig. 7. Dynamic services orchestration models approach

element will be transformed by tagging parameters. The *match* operation is realized before the transformation one. The match binds and checks the concerned (marked) elements that will be transformed. It is also responsible for determining if a parameterable element is compatible with another one. Semantical interpretation among these elements can be supplied by the ontologies use. Our context meta model is ontology supported by the RDF interpretation. According to W3C RDF Semantic a RDF logical semantic is identified by a *triple(x,y,z)* where *x* and *z* are semantical elements, data types or resources in our case (represented by a string), and *y* is the relation between them. The context can be represented by N-triples in URI references. Nevertheless, previous research work propose a semantic solution for context representation; Aspect based services are not taken into account. Dynamicity and adaptability have to be added. The Match class, as illustrated in Figure 8, is responsible for navigating over models. The OCL Rules class specifies the navigation rules using OCL. OCL permits filter expressions to add platform requirements and context information. The *match* operation generates the correspondences between the elements of the Parameterable Element and its correspondents into the Template Parameter. This can be realized by the use of the new SQL queries supported by OCL 2.0. OCL owns a set of types and operations defined in its OCL Library. Some of the types are integer, string, real and boolean. Although, OCL is easily adapted for new types insertion and provides mechanisms for language extension. For example, the *let* expression permits definitions of variables and expressions. OCL also allows attachment of the new variables defined to a method or property. In [26] we defined extensions with the presence of the *match* operation. As aforementioned, the match operation checks the correspondence of the elements evolved in the parameterized transformation. The return value can be a type, property or N-triple. The match navigates over the model searching the marked elements and their correspondences. We are convinced Aspect Based Services Weaving refers to related works in dynamic models composition. We know many questions are till now unresolved as :

- The complete formal description of our meta models.
- The complete specification of our model transformations including context parameters.
- The study of the dynamic models composition including CPIM and AM.

- The dynamicity and event modeling with sequence diagrams, event based modeling.
- The temporized automatons to be generated from BPEL/BPMN as in our previous research works.
- The prototype on a genuine models composition platform.

Let us see now related works.

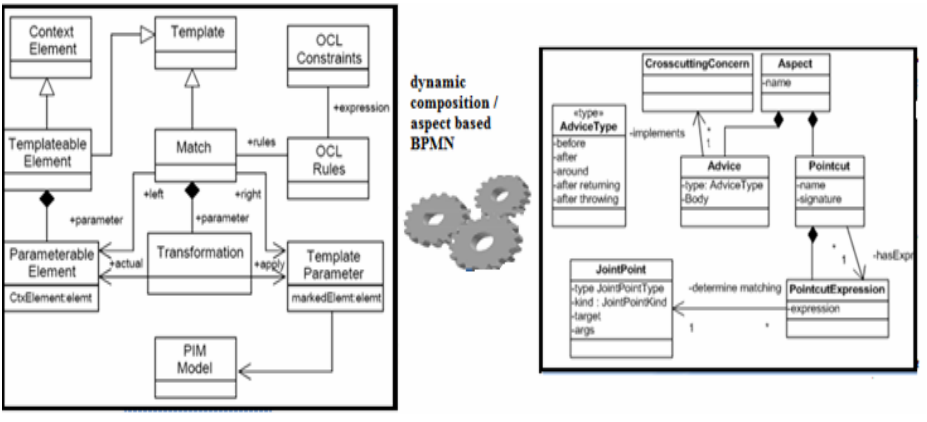

Fig. 8. Parameterized Transformation Metamodel

4.4 Related Works and Discussion

We noticed several related research works and also very interesting approaches to consolidate our work.

For instance [19], have a simpler approach as us. They notice behavioral meta modeling languages were fitted and powerful to perform aspects weaving. They define a semantic-based aspect weaving algorithm for Hierarchical Message Sequence Charts (HMSCs). The algorithm proposed uses a set of transformations that take into account the compositional semantics of HMSCs to weave an initial HMSC and a behavioural aspect expressed with scenarios and with UML sequence diagrams. Other research works about reactive objects, as Hennicker and al [16][17], allow to model synchronous and asynchronous messages by making a clear distinction between inactive (stable) states and activations occurring in different activation phases that follow as a reaction to synchronous operation call. His approach is activity-driven in the sense that during the transformation process different activations occurring in different scenarios but following the same incoming message (in the same state) are integrated into one single activity which models the behavior of an operation across many scenarios. For the integration of the asynchronous scenarios different strategies have been proposed and future work aim to merge these approaches. Context can be stored and recovered as stored variables. This approach seems to be interesting to express services orchestration and context recovering.

[14] notices in software engineering everything evolves very fast: user requirements, technologies, methodologies and applications. Software Product Lines (SPL) modeling technology together with source-code generative tools seem fitting to manage diverse environments with complex, constantly changing relationships. In the context of SPL, they propose an approach based on SmartModels, They propose a meta model to describe business models and a mechanism to compose them. One of the originalities of the meta model is that the designer of the business model can use descriptions of generic entities with a genericity degree which is defined during the model design. Meta-Object Protocol (MOP) which lays the foundation of SmartModels is a mechanism to fill the gap between the semantics and the reification of a model entity. SmartModels may also use AOP paradigm. AOM (Aspect Oriented Modeling) [21] is used to compose models introducing cross cutting concerns in business models to generate with mapping rules aspects based Java code. The aim is to simplify change management with models composition. These research works may be possibly taken into account in the future evolutions of our approach.

5 Conclusion and Future Works

Service-oriented architecture brings new perspectives not only to software architecture but also to enterprise business processes. SOA promotes the use of loosely coupled services to automate business processes. The automation of business processes raises several challenges for enterprises. One of those challenges relates to the how to adapt existing business processes, possibly even at run-time. Thus, changing a collaborative business process can have an impact on the contract specified between the parties involved. Thus, a business process may need to adapt to meet a new contract. We have proposed two different approaches aiming to support service and business process adaptability; however, they both suffer from two important weaknesses:

- Aspect oriented approach, even pragmatic and efficient solution is too complex for non expert users and developers, because it requires strong technical knowledge.
- Context_aware model driven development, using parameterized transformation techniques, is suitable for a contextualization of a service model defined as a PIM. However, it doesn't take into account service orchestration which deals more with a dynamic part and interactions between services.

To overcome the above mentioned limitations, we propose in our future work to mix the two approaches. We propose a Model Driven Development Context_Aware Service Aspect approach with the following features:

- Context modeling allows providing information and situation which intervene in the process of service adaptability.
- Services are unaware of their context and the aspects adapt them to the current environment according to the current context. Context-dependant behaviors are extracted into aspects and weaved with the base service during execution.
- Using model driven development, context models are built as independent pieces of application models and at different abstraction levels then attached by suitable transformation techniques.

- Parameterized transformation technique allows binding context information to a service at a model level, and therefore, which aspect should be weaved at execution level.

We are developing CPSM part and we are working now on the mapping rules definition.

References

[1] Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)

[2] Andrews, T., et al.: Business Process Execution Language for Web Services. 2nd public draft release, Version1.1 (2003), http://www.ibm.com/developerworks/

[3] Baligand, F., Monfort, V.: A concrete solution for Web Services adaptability using policies and aspects. In: ACM-International Conference on Service Oriented Computing (ICSOC), New York, USA (2004)

[4] Tidwell, D.: Web services: The web's next revolution (2000)

[5] Charfi, A., Mezini, M.: Aspect-oriented Web service composition with AO4BPEL. In: Zhang, L.-J., Jeckle, M. (eds.) ECOWS 2004. LNCS, vol. 3250, pp. 168–182. Springer, Heidelberg (2004)

[6] Clark, J., DeRose, S.: XML path language (xpath) ver. 1.0 (1999), http://www.w3.org/tr/xpath

[7] Strang, T., Linnhoff-Popien, C.: A Context Modeling Survey. In: First International Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp (2004)

[8] De Farias, C.R.G., Pires, L.F., van Sinderen, M.: A MOF Metamodel for the Development of Context-Aware Mobile Applications. In: Proceeding of the 22nd ACM Symposium on Applied Computing SAC 2007 (2007)

[9] Tigli, J.Y., Lavirotte, S., Rey, G., Hourdin, V., Cheung-Foo-Wo, D., Callegari, E., Riveill, M.: WComp Middleware for Ubiquitous Computing: Aspects and Composite Event-based Web Services. Annals of Telecommunications 64(3-4), 197 (2009)

[10] Staab, S., van der Aalst, W., Benjamins, V.R.: Web services: been there, done that? IEEE Intelligent Systems [see also IEEE Intelligent Systems and Their Applications] 18(1), 72–85 (2003)

[11] Ferrara, A.: Web services: a process algebra approach. In: ICSOC 2004: Proceedings of the 2nd international conference on Service oriented computing, pp. 242–251. ACM Press, New York (2004)

[12] David, F.S.: Model Driven Architecture: Applying MDA to Enterprise Computing. Wiley Publishing, Inc., Chichester (2003)

[13] Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M.H., La Rosa, M.: Configurable Workflow Models. International Journal of Cooperative Information Systems, IJCIS (2008)

[14] Ţundrea, E., Lahire, P., Pescaru, D., Chirila, C.B.: SmartModels — an MDE platform for the management of software product lines Automation, Quality and Testing, Robotics. In: IEEE International Conference on AQTR 2008, May 22-25, vol. 3, pp. 193–199 (2008)

[15] Haddad, S., Moreaux, P., Rampacek, S.: Client synthesis for web services by way of a timed semantics. In: Proceedings of the 8th Int. Conf. on Enterprise Information Systems (ICEIS 2006), pp. 19–26 (2006)
[16] Henricksen, K., Indulska, J., Rakotonirainy, A.: Modeling context information in pervasive computing systems. In: Mattern, F., Naghshineh, M. (eds.) PERVASIVE 2002. LNCS, vol. 2414, p. 167. Springer, Heidelberg (2002)
[17] Hennicker, R., Knapp, A.: Activity-Driven Synthesis of State Machines. In: Dwyer, M.B., Lopes, A. (eds.) FASE 2007. LNCS, vol. 4422, pp. 87–101. Springer, Heidelberg (2007)
[18] Hmida, M.B., Tomaz, R.F., Monfort, V.: Applying AOP concepts to increase web services flexibility. Journal of Digital Information Management (JDIM) 4(1), 37–43 (2006)
[19] Klein, J., Hélouet, L., Jézéquel, J.M.: Semantic-based weaving of scenarios. In: Proceedings of the 5th International Conference on Aspect-Oriented Software Development (AOSD 2006), Bonn, Germany. ACM, New York (2006)
[20] Kiczales, G., Lamping, J., Maeda, C., Lopes, C.: Aspect-oriented programming. In: Aksit, M., Matsuoka, S. (eds.) ECOOP 1997. LNCS, vol. 1241, pp. 220–242. Springer, Heidelberg (1997)
[21] Lundesgaard, S., Solberg, A., Oldevik, J., France, R., Oyvind Aagedal, J., Eliassen, F.: Construction and Execution of Adaptable Applications Using an Aspect- Oriented and Model Driven Approach. In: Indulska, J., Raymond, K. (eds.) DAIS 2007. LNCS, vol. 4531, pp. 76–89. Springer, Heidelberg (2007)
[22] Tomaz, R.F., Hmida, M.B., Monfort, V.: Concrete solutions for web services adaptability using policies and aspects. The International Journal of Cooperative Information Systems (IJCIS) 15(3), 415–438 (2006)
[23] Monfort, V., Hammoudi, S.: On the Challenge of Adaptable SOA: Model Driven Development, context and Aspect Oriented Programming. In: Proceedings of the second International conference on Web and Information Technologies, ICWIT 2009, ACM SIGAPP, June 12-14, Kerkennah Island Sfax Tunisia (2009)
[24] OMG. Model driven architecture. Document ormsc/2001-07-01 (2001)
[25] OMG. QVT-Merge Group. Query, View and Transformations for MOF 2.0. OMG (2005)
[26] Vale, S., Hammoudi, S.: Model Driven Development of Context-aware Service Oriented Architecture. In: PerGrid 2008, São Paulo – Brazil, July 16-18 (2008)