Using Dynamic Composition And Model Transformation To Generate A Context Aware Platform

Yosra Mlouhi1,2, Valerie Monfort2, Harabi M.Amine3

1SOIE, ISG Tunis, 41 Rue de la liberté, Cité Bouchoucha 2000, le Bardo, Tunis, Tunisia
2University of valenciennes, LAMIH, CNRS, UMR 8201, Valenciennes, France
3Time université 45 Avenue Mohamed V -Montplaisir 1002 Tunisia

Abstract— WComp is a context aware middleware which federates three main paradigms such as: 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 concepts allow component-based compositional approach to design higher-level composite Web Services and incrementing the graph of cooperating services for the applications. However, this embedded and sophisticated middleware can appear complex to neophytes. In previous research works, we proposed to abstract this complexity and to use Model Driven Development which is as an approach for context-aware application development. After a first experience, PSM (Platform Specific Model) has to be refined and lightened particularly. Firstly, we propose a two layers PSM for WCOMP such as: concrete and abstract layers. Secondly, we compose abstract layer with context. Thirdly, we use transformation rules from abstract to concrete PSM. We illustrate our approach with a concrete example.

Keywords-component; Context Aware Applications, Model Driven Development, Service Platform, Adaptability, Platform Specific Model.

Introduction:

As envisioned by Weiser [10], pervasive computing, where computers are everywhere and a person interacts with portable devices that are sensitive and responsive to him, has become a reality nowadays. Pervasive computing is characterized by constant changes in the environment, often caused by the mobility of the users. Context-aware mobile applications can capture dynamically and take advantage of contextual information (such as user location, time and weather, device and user activities). Thus, context-aware applications can sense their context of use, and adapt their behavior accordingly. Context adaptation platforms such as WCOMP [14], OpenORB [15], Aura [17], Cortex [16], and OpenCom [15] aim to manage contextual data. However, these solutions remain complex technically [7], and we propose to raise the level of abstraction using Model Driven Development (MD D) [8]. A first prototype allowed us to test our approach defined in [7] and to improve it as shown in this paper. In this paper we focus on context aware platform such as WComp which is the chosen platform specific model (PSM).

We define: i) an approach to dynamically compose and transform models used to express WComp and generate platform code with transformation rules, ii) a consensual context model, iii) an abstract view of the contextualized platform specific model (CPSM) as the result of the dynamic composition at runtime between context model and WComp model, iv) a concrete view of the platform specific model, v) transformation rules.

The remaining of this article is structured as follows: Section 2 presents the technical background of this paper, section 3 explains the main concepts of MDA (Model Driven Architecture), section 4 shows our contribution, section 5 briefly describes related works, and finally, in section 6, we conclude.

I. TECHNICAL BACKGROUND

A. Overview

Context-aware applications can capture dynamically and take advantage of contextual information (such as user location, time and weather, device and user activities). Various definitions are given and summarized in [2, 3, 6, 7]. In [11] the author limits a context to “a set of information, which is structured and shared. The definition proposed in [3] also presents the context as being hierarchically organized and differentiates between environmental information that determines the behavior of mobile applications and which is relevant to the application. However, it is difficult to give a complete definition for a context. The notion of context is not universal but relative to some situation and application domain [2]. Basically, it should answer the following questions “who?”, “what?”, “when?” and “where?” and, ideally, should allow the system to answer one last question: “why?”. Many research works aim to define Context Aware Platforms and their interoperability [18]. In previous research works we proposed a detailed classification and comparison between platforms, and we selected WComp for its benefits such as event-based Web services, dynamic reconfiguration, containers and Aspect Assemblies.
B. **WComp**

[14, 15] propose a middleware approach called WComp which federates three main paradigms such as:

- Event-based Web services paradigm: based on composite services, which are services whose implementation calls other services and basic services, whose implementations are self-contained and do not invoke any other services. They are generally Web services for devices like UPnP or DPWS. These services handle heterogeneity, extensibility with the reuse of composite services, scalability, security using Web services, etc...

- Lightweight component-based paradigm inside composite Web services: a composite service is based on an internal lightweight components assembly to manage composition between other event-based Web services and to design the interface of a new higher-level composite service. This paradigm is called service lightweight component architecture (SLCA), which is based on events, and a minimum of extra-functional properties unlike SCA. A composite service is then a WComp container managing a dynamic assembly of lightweight WComp components and providing an event-based Web service interfaces. Components handle the high dynamism of the model, providing a way to be structurally adapted. They also address reactivity, since they use event-based communications. A composite event-based Web service is dynamically managed using an internal lightweight components assembly.

- Adaptation paradigm using the original concept called Aspect of Assembly (AA): this concept allows preparing kinds of independent and crosscutting schemes of adaptation dealing with separation of concerns, logically mergeable in case of conflicts and applicable to every composite Web service of the application, not necessarily known (a priori). Aspects provide adaptation to the model, which is structural, since the internal component assembly of composite services is modified without modifying black-box base components. Adaptations, as a set of AA, are designed without knowing event-based Web services of the applications. They are applied (weaved for AA) to the set of event-based Web services of the applications at runtime implementing required adaptations.

However, we noticed these advanced mechanisms are very complex for neophytes and we proposed to abstract PSM with MDA based approach.

II. **MODEL DRIVEN ARCHITECTURE (MDA)**

A. **Presentation**

The MDA (Model Driven Architecture) approach is the initiative of the OMG [8], which introduced the notion of PIM (Platform Independent Model) and PSM (Platform Specific Model). A PIM is a model of a system that concentrates only on the business logic of the application and contains no technical details. A PSM, on the other hand, is a representation of the same system containing all technical details necessary to realise it on a concrete technology platform. The mapping between PIM and PSM is realised using (semi-)automatic transformations [9]. Nowadays, it is well recognised that the process of model transformation is one of the most important operations in MDA. Fig. 2 presents the most common scenario of these transformations, which is compatible with the MOF2.0/QVT standard. Transformation rules specify how to generate a target model (i.e. PSM) from a source model (i.e. PIM). To transform a given model into another model, the transformation rules map the source into the target Meta model. The transformation rules are based on a transformation language, such as the standard QVT. The transformation engine takes the source model, executes the transformation rules, and produces the target model as output.

![Fig. 1 Weaving process in WComp.](image)

Mapping [9] is a mechanism used to interconnect models items to perform semantic equivalences. The aim is...
to achieve the creation of a target model from a current model. Fig.3 shows current and target models with specific correspondences. On one hand, the circle shows a correspondence item identified by a name. On the other hand this graphical notation shows the composition of correspondence items.

![Fig. 3 Correspondences.](image)

[4] [9] present composition such as an operation combining several models via one model, new links and new artifacts may be generated. Model composition is an integration process using several input models to apply a composition operation to generate a composite output model. Research works propose different tools to support dynamic composition. Some implement dynamic composition at design time (such as AMW, Kermeta ) other ( such as DiVA [1] (Dynamic Variability in complex, Adaptive systems) proposes dynamic composition at design time and at run time. Moreover, DiVA intends to devise an approach based on Aspect-Oriented Software Development (AOSD) and Model-Driven Development (MDD) techniques. We decided to use DiVA for the composition step.

**B. The approach**

We use these techniques to propose automatic code generation from one PIM to the PSM of a context aware platform such as WComp. First of all, we are convinced 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 based services (ABS) model is derived. This aspect model specifies the behaviors linked to the context model. Composition is dynamic, on run time, and allowed by DiVA Platform [1]. Fig.4 illustrates the main models and transformations techniques involved in our MDD approach [2]. Five main objectives are illustrated: i) A separation between context information (CM) and business logic (PIM) in individual models, ii) The derivation of an aspect based service model (AM) from a context model. A context model specify the contextual entities with their properties (static view), while the service based aspect model specify behaviors (dynamic view), iii) 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, iv) The Weaving process add adaptability mechanism producing a CPSM model, v) Finally, a CPSM model is mapped into a service platform for future execution of context-aware services.

However, after experimentation with WComp, this approach shows some lacks such as: i) a concrete methodological approach to define each model, ii) a way to validate and to test (models, composition, transformation rules, etc), iii) the gap between modeling process and DiVA process, iv) a manner to lighten the CPSM modeling step.

Context Aware Platforms are generally based either on Aspect either on reflexion paradigms. They al so use Web services to promote interoperability. We are convinced CPSM is divided into two main abstraction levels such as: an abstract PSM and a concrete PSM (Fig.4). Abstract PSM expresses specific concepts used to define the platform composed to context and (aspect based) services. This composition is dynamic on run time and uses DiVA toolkit. For instance, in abstract PSM we can find following concepts such as: Aspect Assembly, link, port, message, service, etc. The result of dynamic composition is a model called “abstract contextualized and service based PSM”. Secondly, a concrete CPSM is modeled. We recommend to re engineer the code of the context aware platform and to study the instantiation of the Aspect Assemblies (and other concepts) with case studies. In concrete CPSM we have: bean proxy, UPnP proxy, containers, libraries, etc. Then, we use mapping techniques and transformation rules. The benefits of this approach is to separate concepts of the platform and its concrete implementation. For instance, WComp can be supported by Java or .Net framework. So, this approach allows switching from Java to .Net easily. In following section, we propose two Meta models abstract PSM and concrete CPSM, in order to illustrate the transformation technique allowing the weaving of context information with technical modelling.

**III. ABSTRACT PSM AND CONCRETE CPSM**

**A. The general approach**

Fig. 5 shows the abstract vision of WComp PSM with the different concepts required for dynamic reconfiguration. Containers encapsulate different beans such as Web services, UPnP, etc, which own ports allowing communicating. On design time, user defines beans and orchestrations (links) by configuring ports. On run time, according to context and events providing from context model instantiation, links are generated, and the fitted assemblies configurations are loaded and executed.

**B. Abstract PSM**

No consensus was found concerning context model definition. So, we proposed our own solution. Fig. 6 illustrates our context Meta model around the main entity
“context view” which is composed by a set of “context entities”. In our Meta model, a business service communicates with messages that may handle context entities. The Web service will be the implementation of the business service. The entity, “Contextual Entity” specialized in three main classes: Actor, Environment and Computing Entity. Actor can be a person or another object that has a state and profile. It evolves in an environment and uses computational devices to invoke services. The state of an Actor can be “moving” or “fixed”. Computing Entity is the mobile computing system used by the actor to access the services and capture contextual information from the environment. The computational device can obtain information concerning the type of device it is (PDA, laptop, cellular phone...), the application, the network, etc. Environment is constituted of all information surrounding the actor and its computational device and that can be relevant for the application. It includes different categories of information such as: i) Spatial context information can be location, city, building, etc, ii) Temporal context information comprises time, date, season, etc, iii) Climate can be temperature, type of weather. The environment includes environment constraints as “sound is null” and environment rules as “if temperature is lower than 10 then do transaction”.

We aim to compose abstract PSM with context, aspect, and Web services models. As shown Fig.6, we defined the Basic Assembly class which is aligned with the Environment class from the context model. It means a basic assembly is sensible to the environment. The Context entity will be linked to the service class to indicate that service communicates with the contextual entity with one or more messages. The Actor class is aligned with the class Aspect Assembly (AA) from a context model to illustrate that actor can select one or more AA.

C. Abstract Contextualized PSM (CPSM)

No consensus was found concerning context model definition. So, we proposed our own solution. Fig. 7 illustrates our context Meta model around the main entity “context view” which is composed by a set of “context entities”. In our Meta model, a business service communicates with messages that may handle context entities. The Web service will be the implementation of the business service. The entity, “Contextual Entity” specialized in three main classes: Actor, Environment and Computing Entity. Actor can be a person or another object that has a state and profile. It evolves in an environment and uses computational devices to invoke services. The state of an Actor can be “moving” or “fixed”. Computing Entity is the mobile computing system used by the actor to access the services and capture contextual information from the environment. The computational device can obtain information concerning the type of device it is (PDA, laptop, cellular phone...), the application, the network, etc. Environment is constituted of all information surrounding the actor and its computational device and that can be relevant for the application. It includes different categories of information such as: i) Spatial context information can be location, city, building, etc, ii) Temporal context information comprises time, date, season, etc, iii) Climate can be temperature, type of weather. The environment includes environment constraints as “sound is null” and environment rules as “if temperature is lower than 10 then do transaction”.

We aim to compose abstract PSM with context, aspect, and Web services models. We defined the Basic Assembly class which is aligned with the Environment class from the context model. It means a basic assembly is sensible to the environment. The Context entity will be linked to the service class to indicate that service communicates with the contextual entity with one or more messages. The Actor class is aligned with the class Aspect Assembly (AA) from a context model to illustrate that actor can select one or more AA.

D. Concrete CPSM

Two kinds of files are generated .cs and .dll files. In .cs file a main owns a function which launches Container creation and starts application. Container owns a method to create dynamic link between beans for orchestration and dynamic reconfiguration. One or several libraries in .dll files are invoked for assembly and instanciation

E. Mapping and tranformation

We proposed an abstract PSM of WComp we dynamically composed, with DIVA tool, with context and aspect based services. Then, we define a concrete Contextualized PSM from a reverse engineering of WComp code. The last step of our approach is the mapping and transformation rules definition. Then, we propose a case study to illustrate our proposal.

IV. APPLICATION

A. Case study

Intelligent Houses are very similar to Programmable houses that aim to perform some activities such as opening a door or switching the light, etc. We consider an intelligent house that can detect devices and provide services to users such as: i) Thanks to sensor systems and RFID, a lamp can dynamically be illuminated according to room lighting and when the user comes in the room. So the system detects user’s presence with a GPS system and luminosity with appropriate sensors, ii) On the other hand, if the user enters a room and if the door is closed it is sufficient for user to approach the door so that it opens, iii) In addition, if the user wants to watch TV, if it is in the right part of room they simply say “TV” to turn on with a voice recognition system.

B. Context and BPMN Models

The following class diagram is an extract of the previous model proposed in Fig. 10 according to the scope of the case study. An interceptor is responsible for intercepting presence and position of user. An interceptor can be: a sensor, a sound detector or a light detector. After modeling class diagram, we model the business processes with BPMN formalism in Fig. 7.
Fig. 5 Dynamic Composition and Transformation Approach

Fig. 6. Abstract PSM model
C. Dynamic composition with DiVA

DiVA offers an interface to describe functional features and feature Tree Extraction from textual documents. The extraction functionality must be applied on a plain text (*.txt) file. It is invoked by selecting the ArborCraftExt option from the right-click menu on the text file. This leads to an interface requesting definition of a number of parameters. Feature models can be converted quickly and accurately to a DiVA model representation. The conversion must be carried out upon completion of the feature model analysis at the requirements level by selecting the feature model and invoking the FMP2DiVA tool for it, as shown in Fig. 8. The result of this conversion is a DiVA WP2 projects populated with the details of feature models, variability, constraints, and context elicited during the requirements engineering task. Fig. 8 shows the context requires reconfiguration of the system with: the presence of the user, the light level (low, high), the positioning of the user (near the door, in the room), the validation of the vocal message, and a voiceprint of the user. ii) The functional properties of the system can be optimized while the system performs adaptation.

The basic model modeled with the ART approach (at Run-Time Architecture)[1], includes the general architecture of the application and the components that are included in all system configurations. Fig. 9 shows the basic model of the case study that represents the system such as component instances indicating transmission binding (connections between the components through the corresponding ports), components primitives which contain ports and services provided by the system. The utilization of components instances ensures the initial component cannot be modified.

Each modification defined in adaptation models is represented in a smart model which will be woven in reconfiguration is required to ensure adaptation. This model is described by the SmartAdapter approach. Indeed, the aspect model includes three parts such as: point cut model, advice model and a composition protocol, as shown in the following codes:

D. Generated code example

So, Code 1 describes a model of point cut. This architecture fragment specifies the place where aspects has to be weaved. The aspect model "lampe_aspect" indicates where this aspect is woven. Line 4 expresses the instance of the intelligent_house where is weaved the light component (line 9) with the required ports: presence_user (line 12) and luminosity (line 13) to provide the service...
lampe_switching (line 7) by the lampe_switching port (line 14).

Code 1. Point cut of the "lampe_aspect" aspect model

When aspect is defined it is compiled to generate a .DRL file which is used to weave the aspect in a basic model. During execution, some sensors are deployed to control environment or user requirements changes. These events are provided to a rule engine which decides if reconfiguration is necessary or not, what is the new set of variants that are required to fit to the new context.

E. Related Works

Many research works based on MDA aim to define PIM and PSM. Some approaches [9] integrates aspects, dynamic composition, business models for PIM, and, transform PIM into PSM. No research work aim to model context aware platform such as WComp. [14] provides a kind of WComp meta model with SLCA, but it is not complete and it mixes several concepts such as aspects, Web services, etc... included in one model without compositional approach.

V. Conclusion

This research work aims to propose a manner to generate the code of an embedded context aware platform called WComp. The approach is based on MDA, transformation rules, concepts mapping between PIM and PSM, and dynamic composition based on DiVA platform. It allows describing each step and focuses on PSM part by refining it with Abstract PSM composed with context model, and Concrete PSM. We successfully applied this approach in other domain such as M-Learning based on mobile agents and Multi agent systems. PIM models M-Learning domain and PSM includes mobile agents’ concepts as abstract PSM and Multi agent systems as concrete PSM. Context includes location and learner profile. We aim to improve transformation rules, to iterate to get correct result and to test transformation rules and composition. We also aim defining a genuine method to model PIM and PSM and generate code including dynamic testing.

REFERENCES

doi: 10.1017/S0269889907001208