MODEL DRIVEN DEVELOPMENT OF CONTEXT-AWARE SERVICES USING PARAMETERIZED TRANSFORMATION

Slimane Hammoudi
ESEO 4, Rue Merlet de la Boulaye, B.P. 9249 009, Angers Cedex 01, France

Keywords: Context Aware Services, Model Driven Development, Service Platform, Parameterized transformation.

Abstract: Context-aware development has been an emergent subject of many research works in ubiquitous computing. Few of them propose Model Driven Development (MDD) as an approach for context-aware application development. Many focus on context capture and adaptation by the use of legacy architectures and others artefacts to bind context with application logic. This work proposes a new approach called COMODE (Context Aware Model Driven Development) which advocates Model Driven Development to promote reuse, adaptability and interoperability for context-aware application development on service platforms. In this paper we focus on the transformation issue and propose a parameterized transformation as a new approach for model driven development of context-aware services.

1 INTRODUCTION

Traditional computing applications are often static and inflexible. They are designed to run on a specific device, offer a number of predetermined functions to the user and have contextual dependencies embedded in them. Such application models are not suited to operate in a pervasive computing environment, which is characterized by richness of context, by the mobility of users and devices (PDAs, smartphones,…) and by the appearance and disappearance of resources over time (Vukovic, 2004). Nowadays, where ubiquitous (pervasive) computing, based on context-awareness, takes a very important place in daily life, it becomes necessary to develop context-aware applications providing adequate services for the users by taking into account their multiple contexts. The design of these pervasive applications which must adapt to different contexts (Dey, 2001) can be developed using Service Oriented Architecture (SOA). Indeed, the loose coupling and interoperability inherent to SOA may provide context-aware services (Grassi, 2007). Based on the SOA paradigm, various research works for the development of context-aware services were carried out by proposing different approaches and methodologies (Kapitsaki, 2009). However, in most of the approaches there is a lack of generic methodology for formalizing the development activity for this type of services, thus making it very cumbersome and time consuming. Recently, some research proposals have advocated Model Driven Development (MDD) as an approach for context-aware services development (DeFarias, 2007). MDD allows the development of services by separating context information from business logic in a set of different abstraction model constructions and by using different transformation techniques. MDD has many important benefits as concerns separation, reuse of models and interoperability (Vale, 2008). In this paper we propose a new approach called COMODE which advocates Model Driven Development to promote reuse, adaptability and interoperability for context-aware application development on service platforms. We focus our discussion on the transformation process and we propose a parameterized transformation technique to weave context information with business logic at model level. The remaining of this article is structured as follows: section 2 presents our proposed approach COMODE through its architecture, section 3 presents the transformation process and Section 4 illustrates our approach through an example of context-aware mobile service. Section 5 concludes this paper and discusses the perspectives.

2 COMODE: PRINCIPLE AND ARCHITECTURE

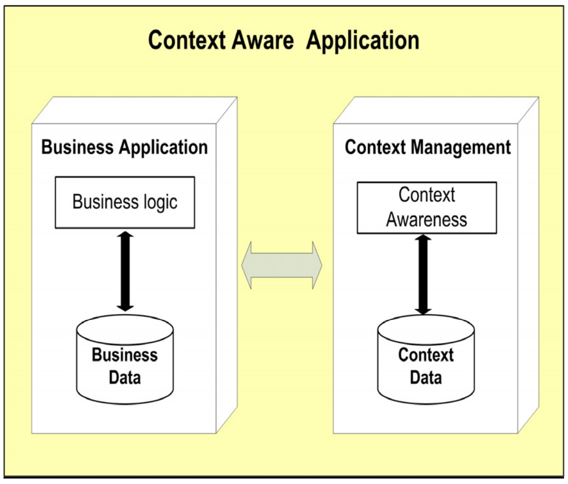
Our approach COMODE is based on the principle of separation of concerns, and emphasizes this principle during the design of context-aware application. The different concerns are represented by independent models. Figure 1 illustrates on the one hand the principle and the main components of our approach (a) and on the other hand the COMODE architecture based on five views (b).

Our five architecture views are inspired from the standard EDOC-ECA (OMG, 2004) views that in turn are directly taken from the Reference Model of Open Distributed Processing RM-ODP (ODP, 1995) The EDOC-ECA views are: the Enterprise view, the Computational view, the Information view, the Engineering view and the Technology view. More details concerning these views are presented in (ODP, 1995). We have adapted these views to treat the different requirements of the development of distributed context aware applications, in an independent way i.e., context definition, representation, adaptation, reasoning and binding. According to figure 1 (b), the Business, Context and Composition Views are expressed by PIM models, i.e., these models abstract platforms details.

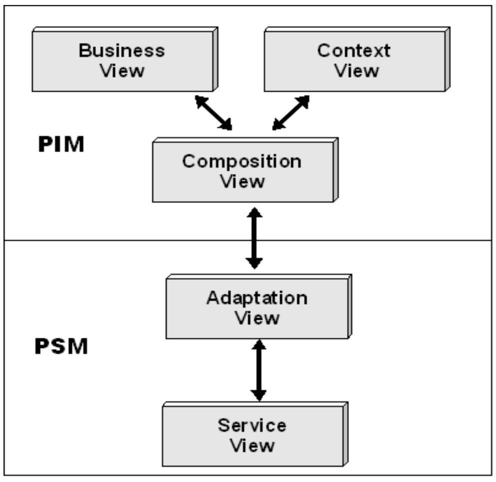
The Business View is based on the Enterprise Viewpoint and it focuses on business logic, roles and activities. The Context View is based on the Information Viewpoint and it is responsible for defining context information, capturing, representation and interpretation. This View is also responsible for defining context-aware elements, i.e., software elements that realize activities based on context information (components and operations). The Composition View establishes the link between Business logic and Context logic, i.e., how business statements and contextual activities will be executed harmoniously to provide relevant results for user needs. The Adaptation and Service Views are expressed in PSM models. They are based on the Engineering Viewpoint, i.e., they specify target platforms details, middleware technologies, connectors, protocols and others platform specific requirements. These views are specific to an abstract platform but they are not dependable of a particular platform (Service platform versus J2EE platform). In this paper we are concerned by the composition view and we will discuss in the next section a transformation mechanism to integrate the context into the business logic of an application using a model driven approach.

3 TRANSFORMATION APPROACH

The separation of concerns (business and context) is emphasized at the model level of our approach where PIM and context models are defined independently, and then merged by suitable transformation techniques. Two types of transformations are involved in our proposal. The first type of transformation called Parameterized transformation allows weaving context information with business logic at model level. We have investigated (Vale, 2008); (Monfort, 2009)



(a): Principle and the main components



(b) Five architecture views

Figure 1: Main components and architecture of a context-aware application in COMODE.

this type of transformation which is not sufficiently explored nowadays and for which there is no standard transformation language. 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. The second type of transformation is the traditional transformation technique using a language such as QVT, which operates in two steps, mapping specification followed by transformation definition. Parameterized transformation is based on parameter paradigm (Frankel, 2003) (Vale, 2009). The OMG (OMG, 2001) has defined the concept of parameter as follows:

"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 (Frankel, 2003), 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. Transformation by parameter could be used to improve new functionalities (values, properties, operations) or to change the application behavior (activities). In our approach, we are convinced that parameterized transformation focusing on PIM to PIM transformations is the fitted solution. The designer must specify the parameters to be inserted during the transformation phase. In our proposition these parameters represent the context and the transformation process will join this context information into the business application (PIM) 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 engine. The marked elements represent context-aware elements, in others words, the model elements that can be contextualized.

The transformation language must support parameterization. In our case the parameters can be a Context Property and/or a Context Data Type. We use templates to specify which elements in the application model are potentially context-aware. A detailed discussion about our approach is presented

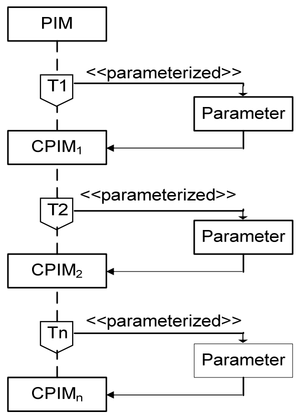


Figure 2: Parameterized Transformation.

in (Monfort, 2011). The transformation engine must navigate through the PIM model to verify the parameters and the elements marked and then make the transformation consisting in an update of contextual properties in the PIM. Template parameter (Vale, 09) is an element used to specify how classifiers, packages and operations can be parameterized. UML 2.0 states 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 parameter for a template, or specified as an actual parameter in the binding of the TEMPLATE (VALE, 09). A context parameter can be expressed as a constraint and compared with the element's signature in the template parameter. This operation is named the matching operation.

4 CASE STUDY

In order to illustrate the parameterized transformation technique allowing the weaving of context information with business logic, we take an example of a service provider named *FunFinder*. Using this service provider, a mobile user can find attractions in a city according to two principles:

- The user specify only the wished service (example: A restaurant)
- The system uses in a transparent manner all the relevant information (context) in order to give to the user the most adapted answers.

Figure 3 shows an example illustrating the parameterized transformation technique. Three models are presented: The business logic model (PIM) of the service provider *FunFinder*, the context model in the middle, and finally the merging model CPIM. In this example we are particularly interested in a restaurant finding for a mobile user.

Initially, in the business model a greeting service is invoked that greets the user with a welcome message in his native language. Then, the user invokes the RestaurantsFinder service to find a restaurant. The findByContext ( ) method whose parameters are defined in a CPIM model, allow to find a restaurant with two contextual parameters: the user culinary preference as well as its location. The context model groups the relevant contextual entities for the RestaurantsFinder service. We thus have an actor which represents the mobile user with in one side his profile limited to his culinary preferences, and in another side, also his mobile phone allowing giving anytime the user location. In order to implement the transformation process allowing to weave the context model and the business logic model, the designers of the two models should update respectively the PIM model and the context model according to figure 9 as follows:

- ✓ In the business logic model, elements that represent context are marked and if necessary a template may precise their generic type. In our example the designer mark the person class as a main contextual entity and use a template to precise the generic type of this class: user.
- ✓ In the context model, all the relevant contextual entities for the *RestaurantsFinder* service are marked. Thus, the actor, its culinary profile and its location are marked. The *template* linked to the actor class allows defining the generic type of this class. The *MobilePhone* class has not been marked as it doesn't impact in the process of finding a restaurant. It's the class location which intervenes in the process of finding a restaurant.

The parameterized transformation process takes as input the two models: PIM and context, and execute a *matching* operation between these models. The *person* class (marked) from a PIM model is aligned with the class *actor* from a context model as they have the same generic type *user*. The alignment process produces a CPIM model composed by all the relevant classes involved in the *RestaurantsFinder* service. The parameters of the method *findByContexte* (FoodPreference and Location) of *RestaurantsFinder* class will be linked to the two classes *FoodPreference* and *Location,* in order to find restaurants according to the data from these two classes.

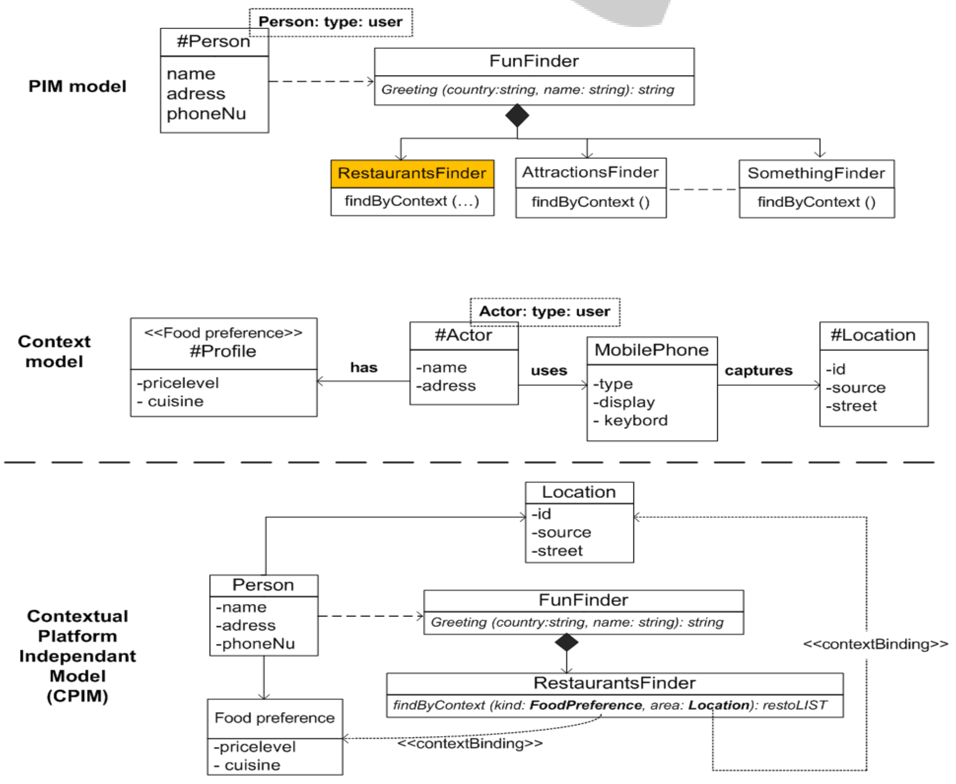


Figure 3: Parameterized transformations for context weaving.

5 CONCLUSIONS

In this work we have investigated the development of context-aware services in a mobile environment and we have proposed an approach called COMODE to develop these services according to a model driven approach. Mobile services are traditional services delivered via mobile devices, such as mobile phones or PDA's. Mobile services can also be specifically tailored to the needs of mobile users. A context-aware mobile service is adapted to the current situation of the user. The goal of a context-aware service is to support the user by providing him with the right service at the right moment. If the user context changes, the context-aware service should self-adapt or be adapted to the new context. A context-aware service is autonomous and tries to support the user without too much interaction with a computing device. Several approaches have been proposed to answer the challenges of mobile and context-aware service development. However, most of these approaches merge context information and context-aware activities with business logic. We promote context independence by the MDD concerns separation. Context models are defined independently of business logic models and context aware statements are defined in individual components developed independently of application ones.

We have proposed a model driven context-aware approach aiming to support service adaptability. The main features of our approach are:

- Context modeling allows to provide information and situation which intervene in the process of service adaptability.
- Services are unaware of their context and the context aware mechanisms adapt themselves to the current environment according to the current context. Context-dependent behaviors are extracted into "context services" 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 techniques allow the binding of context information to a service at a model level, and therefore, allows specifying which behavior should be weaved at execution level.

REFERENCES

De Farias, C. R. G., Pires, L. F., and van Sinderen, M. (2007). A MOF Metamodel for the Development of Context-Aware Mobile Applications. In *Proceeding of the 22nd ACM Symposium on Applied Computing* (SAC'07) pages: 947 - 952.

Dey, A. K. (2001). *Understanding and Using Context. Personal and Ubiquitous Computing* 5, 1, 4-7.

Frankel S. David. (2003). *Model Driven Architecture: Applying MDA to Enterprise Computing,* Wiley Publishing, Inc.

Vincenzo Grassi and Andrea Sindico, *Towards model driven design of service-based context-aware applications*, ACM (2007), pp. 69-74

Kapitsaki, G. M., Prezerakos, G. N., Tselikas, N. D. et Venieris, I. S. (2009). Context-aware service engineering : A survey. *Journal of Systems and Software*, 82(8):1285–1297.

Matthias, B., Dustdar, S., and Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of ad Hoc and ubiquitous Computing,* 2007.

Mary, B., and Patrick, B. (2005). Understanding context before to use it. In 5th International and Interdisciplinary Conference on Modeling and Using Context vol. 3554 of *Lectures Notes in Artificial Intelligence*, Springer-Verlag, pp. 29-40.

Monfort, V., Hammoudi, S. When Parameterized MDD Supports Aspect Based SOA , IJEBR 2011, *International Journal of e-Business Research* (To appear).

Monfort, V., Hammoudi, S. ICSOC, Towards Adaptable SOA: Model Driven Development, Context and Aspect *The 7th International Conference on Service Oriented Computing,* November 23-27 2009, Stockholm, Sweden.

OMG (Object Management Group). (2001*). Model Driven Architecture (MDA),* OMG document number ormsc/2001-07-01.

Vale, S., Hammoudi, S. Model Driven Development of Context-aware Service Oriented Architecture. In *PerGrid'08*, July 16-18, 2008 - São Paulo – Brazil.

Vale, S., Hammoudi, S., Context-aware Model Driven Development by Parameterized Transformation *Proceedings of MDISIS,* 2008.

Maja Vukovic and Peter Robinson, Adaptive, planning based, web service composition for context awareness, *Advances in Pervasive Computing* (2004), pp. 247-252