MODEL DRIVEN DEVELOPMENT OF CONTEXT-AWARE MOBILE APPLICATIONS

An Architecture and a Metamodel

Saber Lajili
ISET, Gafsa and ISIG, Kairouan, Tunisia

Slimane Hammoudi, Olivier Camp
ESEO, 4 rue Merlet de la Boulaye, Angers, France

Mohamed Mohsen Gammoudi
ESSAIT, Université 7 novembre, Tunisia

Keywords: MDD, Context-awareness, Metamodeling, Transformation.

Abstract: The development of context-aware application has been the subject of many research works in ubiquitous computing. Nowadays, there is a lack of generic approaches for formalizing the activity of development for this type of applications, and consequently this activity is very cumbersome and time consuming. This work proposes to use Model Driven Development to promote reuse, adaptability and interoperability in the development of context-aware applications. Through the separation of concerns in individual models and transformation techniques, context can be modeled and adapted independently of business logic and platform details. This paper presents an architecture and a metamodel for model driven development of context-aware applications. The architecture illustrates the different steps and techniques involved in the process of development. The metamodel provides a formal representation of contextual information.

1 INTRODUCTION

As envisioned by Weiser in (Weiser, 1991), 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. In contrast to the more traditional desktop-based computing paradigm, pervasive computing is characterized by constant changes in the environment, often caused by the mobility of the users. In the scope of pervasive computing, a class of applications that has raised increasing interest in the research community is context-aware mobile applications. These applications can capture dynamically and take advantage of contextual information (such as user location, time, weather, device or user activities). Thus, context-aware applications can sense their context of use, and adapt their behaviour accordingly. From an implementation point of view, the emerging technology of web services seems promising for such mobile applications in pervasive environments.

In this paper, we focus on context-aware services. This means, services which use the context to carry out activities and provide relevant information to the user without any human interaction from his part. To develop context-aware services, the following three main issues need to be considered:

- Context identification;
- Context provisioning;
- Context-awareness mechanism.

The first issue is to identify what kind of context information will be used. This will allow for the definition of a model describing the contextual domain in which the application or service is defined. A number of context models have been proposed in the literature, and have been reviewed in (Strang and Linnhoff-Popien, 2004).

The second issue of provisioning concerns the derivation of context information. Due to heterogeneity of context providers, sensor imperfection, quality of context information, and dynamic context environ-

Lajili S., Hammoudi S., Camp O. and Gammoudi M. (2009).
MODEL DRIVEN DEVELOPMENT OF CONTEXT-AWARE MOBILE APPLICATIONS - An Architecture and a Metamodel.
In *Proceedings of the 4th International Conference on Software and Data Technologies*, pages 43-52
DOI: 10.5220/0002254200430052

ments, context provisioning is not trivial (de Farias et al., 2007a).

The third issue deals with the mechanisms that can be used by context-aware services in order to adapt automatically their behaviors according to context information.

In this paper, we concentrate on the first of these three issues i.e.; the identification of context information. A majority of existing context aware service oriented applications model and implement both, business and contextual activities together. This design practice hampers software reuse and context management. Motivated by this problem we propose to apply Model Driven Development (MDD) in context-aware services development. The Object Management Group (OMG) recommends the Model Driven Architecture (MDA) approach as a standard in Model Driven Development and considers models as the best way to represent the structure and the semantic of the concepts manipulated by a system. Through the separation of concerns (business and context), the reuse, adaptability and management of context information can be improved. Context models can be built as independent pieces of application models at different abstraction levels that are then attached by suitable transformation techniques.

The remaining of this article is structured as follows: section 2 presents the concepts of context and context-aware applications; section 3 introduces the Model Driven Development approach; section 4 describes our proposed modeling architecture using the MDA approach to develop context-aware mobile applications; in section 5 we lay out the most recent researches on context modelling; we then present and give a detailed description of our context metamodel; finally we conclude our work in section 6.

2 THE CONCEPTS OF CONTEXT AND CONTEXT-AWARE APPLICATIONS

New technologies, in particular, wireless communications, together with the increasing use of portable devices (smart phones, personal digital assistants - PDA-, laptop ...) have stimulated the emergence of a new computing paradigm: pervasive computing. In fact, we have moved from the desktop computing paradigm to the mobile and ubiquitous computing paradigm.

Pervasive computing firstly introduced in 1991 by Weiser, refers to the seamless integration of devices into the users' everyday life. "Appliances should disappear into the background to make the user and his tasks the central focus rather than computing devices and technical issues."(Weiser, 1991). Computing applications now operate in a variety of new settings; for example, embedded in cars or wearable devices. They use information about their context to respond and adapt to changes in the computing environment. They are, in short, increasingly context aware. The context awareness of such applications is the subject of a recent field of studies in pervasive computing: context-aware systems.

This terminology was discussed in (Schilit and Theimer, 1994) and presented as "software that adapts according to its location of use, the collection of nearby people and objects, as well as changes to those objects over time". Since then, there have been numerous attempts to define context-aware computing.

In (Pascoe, 1998) they define context-awareness as the ability of a program or device to sense or capture various states of its environment and itself. Considering these definitions, a context-aware application must have the ability to capture the necessary contextual entities from its environment, use them to adapt its behavior (run time environment) and finally present available services to the user. In this sense and to describe context-awareness independently from application, function, or interface, (Pascoe, 1998) proposes four features of context-aware applications : (1) Contextual sensing which refers to the detection of environmental states and their presentation to the user; (2) Contextual adaptation refers to the adaptation of the application's behavior to the current context; (3) Contextual resource discovery is the use of context data to discover other resources within the same context; (4) Contextual augmentation in which the environment is augmented with digital data associated to a particular context.

In (Dey and Abowd, 1999), the authors introduce another definition in which they insist on the use of context and the relevance of context information. The authors consider that "a system is context-aware if it uses context to provide relevant information and/or services to the user, where relevance depends on the users task". They explain how to use context and propose a classification of the features of context-aware applications that combine the ideas of (Schilit and Theimer, 1994) and (Pascoe, 1998). They consequently define three categories of features that context-aware applications may support: (1) presentation of information and services to the user; (2) automatic execution of a service; and (3) tagging of context to information for later retrieval.

In order to describe how an application can be context aware, a lot of works are described in

(Matthias et al., 2007) and propose various architectures for context-aware systems. All the works converge to a general architecture composed of five ordered layers presented in figure 1. The complete description of every layer is given in (Matthias et al., 2007).

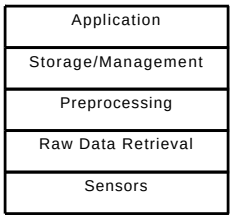

Figure 1: Architecture of context-aware systems.

What is understood by ”context information” in context-aware systems and what could be the definition of context have been the subjects of many recent works. The various definitions of the term are given and summarized in (Mary and Patrick, 2005). The first definitions define specific contexts for specific applications by enumerating concrete contextual entities (Brown et al., 1997; Ryan et al., 1997). For example the authors in (Brown et al., 1997) define context as being information about location, the identity of people in close proximity, physical conditions. In (Ryan et al., 1997), the authors add to this definition the notion of time. Other definitions are extremely broad; the most popular one is given by (Dey and Abowd, 1999): ”Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves”. The authors give a general definition that can be used in a wide range of context-aware applications. To refine their definition, they identify four categories of context that they feel are more practically important than others. These are location, identity (user), activity (state) and time (Dey, 2001).

In (Winograd, 2001) the author approves this definition and claims that it covers all proposed works in context. However he considers it as a general definition that does not limit a context. Thus he proposes his own definition in which he limits a context to ”a set of information, which is structured and shared. It evolves and is used for interpretation”. We stress that the notion of hierarchy (structure) of context introduced by (Winograd, 2001) is important.

The definition proposed in (Chen and Kotz, 2000) also presents the context as hierarchically organized. In this work the authors differentiate between environmental information that determines the behavior of mobile applications and that which is relevant to the application. They thus define the context as ”the set of environmental states and settings that either determines an application’s behavior or in which an application event occurs and is interesting to the user”.

As we have said previously, it is difficult to give a complete definition for a context. In fact, as stated in (Mary and Patrick, 2005), the notion of context is not universal but relative to some situation and application domain. We are interested, in our research, in user centred mobile application. Thus, we consider that the defining context here is a set of information structured in three dimensions:

Actor. A person which is a central entity in our system.

Environment. In which the person evolves and,

Computational Devices. Which are used by a person to invoke services and that captures the different states of the environment.

All the information relative to the three dimensions can also be shared by other mobile applications.

3 MODEL DRIVEN ARCHITECTURE (MDA)

At the beginning of this century, software engineering needed to handle software systems that were becoming larger and increasingly complex. Object-oriented and component technology were insufficient to provide satisfactory solutions to support the development and maintenance of these systems. In order 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. In this new global trend called Model Driven Development, Model Driven Architecture MDA is a particular variant.

3.1 MDA: A Four Layer Architecture

MDA is based on standards from the OMG; it proposes a four layer architecture (OMG, 2001): metametamodel, metamodel, model and information (i.e. an implementation of its model).

Figure 2 presents the basic Metamodeling architecture of MDA and shows the relationships between the different levels of the models. In this approach, everything is a model or a model element.

In level M0, a real system is *representedBy* a model in level M1, and a model in level M1 *conformsTo* a metamodel in level M2. These two important relationships of MDA are discussed in (Bézivin, 2005):

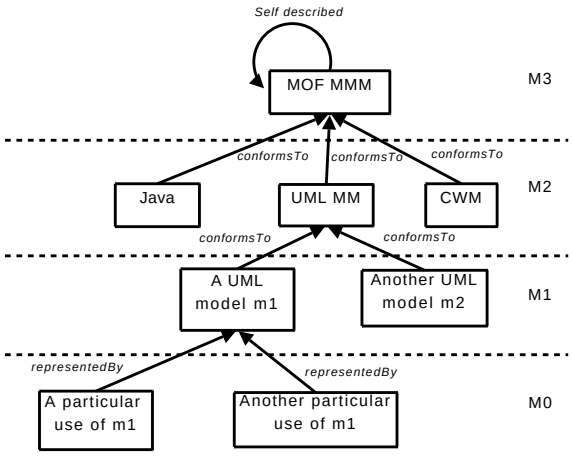

Figure 2: The four meta-layers MDA architecture.

In level M3, a metametamodel is a well-formed specification for creating metamodels such as the Meta Object Facility (MOF), a standard from the OMG. In level M2, a metamodel is a well-formed specification for creating models.

In level M1, a model is a well-formed specification for creating software artefacts. In level M0, an operational example of a model is the final representation of a software system. Besides the four layer MDA architecture, the development is based on the separation of concerns (e.g. business and technical concerns), which are later recombined. Business concerns are represented using the Platform-Independent Model (PIM), whereas technical concerns are represented using the Platform-Specific Model (PSM).

3.2 Model Transformation in MDA

It is well recognized nowadays that model transformation is one of the most important operations in MDA. In the context of the basic four level Metamodeling architecture of MDA, various scenarios of model-to-model transformation have been identified. To transform a given model into another model, transformation rules map a source metamodel into the corresponding target metamodel. The transformation rules are based on a transformation language, such as the standard QVT and are executed by a transformation engine to produce the target model as output.

4 MDD ARCHITECTURE FOR CONTEXT-AWARE APPLICATIONS

Figure 3 illustrates our proposal of an architecture for model driven development of context-aware applications. In section 2, our studies on context representation have conducted us to the conclusion that the most promising assets for context modeling for ubiquitous computing environments and context-aware applications in particular can be found in the ontology formalism.

MDA and its four-layer architecture, presented in section 3, has provided a solid basis for defining metamodels for any modeling language, so it was a straight choice to define an ontology-modeling language using MOF. The Ontology Definition Metamodel (ODM) was recently adopted as a standard by the Object Management Group (OMG, 2006). It supports ontology development and conceptual modeling in several standard representation languages. furthermore, it provides a coherent framework for visual ontology creation based on MOF and UML. The main goal of ODM is to bridge the gap between traditional software tools for modeling (like UML) and artificial intelligence techniques (Description Logics) for describing ontologies. The principle of ODM is to merge Model Driven Architecture and Semantic Web. Basically, ODM allows creating ontologies using UML and transforming them to OWL/RDF, Topic Map or Common Logic. Thus, in our architecture the specification of metamodels (layer M2) involves two formalisms: UML for the definition of the business logic of a given application (PIM Metamodel).

This definition occurs at the highest level of abstraction. The second formalism is ODM from which we define a metamodel for context information using the ontology language RDF. Our metamodel of context information (Ctxt metamodel) groups the main entities and their relationships, found in Context-Aware Mobile Applications. Three types of relationships are involved in our architecture. The *conformsTo* relationship represented by filled arrow and which link a model from layer M1 to its metamodel at layer M2, and a metamodel from layer M2 to a general and unique MOF formalism at layer M3. The *describedBy* relationship represented by the dotted arrow and which specifies the formalism in which a given metamodel is described: UML for the PIM metamodel and ODM for the context metamodel. Finally, hollow arrows reference models that are either used (input) or produced (output) by a transformation process.

The separation of concerns (business and context)

is emphasized at layer M1 of our architecture where PIM and context models are defined independently, then merged by suitable transformation techniques. Two types of transformations are involved in our proposal. The first type of transformation called "Parameterized transformation" represented by ovals, allows merging context information with business logic at model level. This type of transformation, which has been investigated recently (Vale and Hammoudi, 2008), is rarely explored and has no standard transformation language. 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 maps a CPIM model into a CPSM model (contextual platform specific model). As our main application domain concerns mobile applications, Web service platforms seem the most adapted.

Parameterized transformations have been introduced by Frankel (Frankel, 2003), who 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 or to change the application's behavior (activities).

In our proposal, the designer has to mark, in the application model (PIM), 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 thus support such 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 ; This is presented and discussed in (Vale and Hammoudi, 2008). The transformation engine has to navigate through the PIM verifying the parameters and the elements that are marked and then carry out the transformation.

Template parameters (Frankel, 2003) are 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 used as parameters. Such elements would then be exposed as formal parameters for template transformations and specified as actual parameters when binding the template (Vale and Hammoudi, 2008).

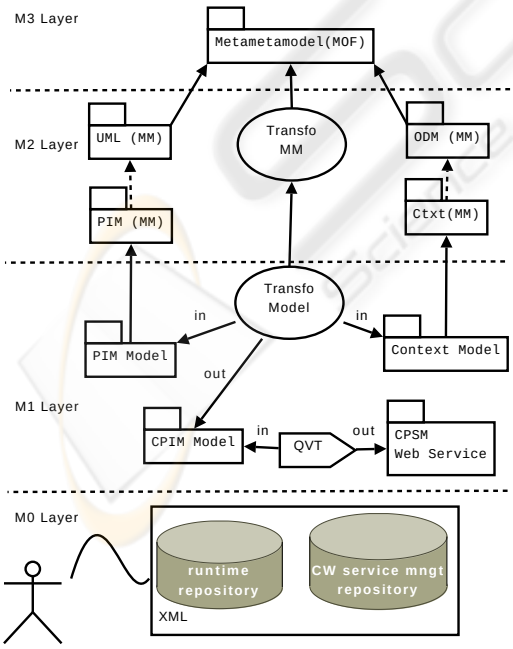

Figure 3: The model driven development architecture.

5 CONTEXT MODELING

To define and store context data in machine processable forms, it is necessary to define and build a context model. The number of new mobile applications that take care of context is growing every year. Thus the task of modeling and designing context seems relevant and constitutes a challenge for researchers. This section investigates a number of context models described in the literature for context-aware mobile applications, and proposes a context metamodel based on the main concepts found on these models.

5.1 Related Work

In the earlier stage of the research, many works focused on building a very specific context aware application such as the Active Badge Location System (Want et al., 1992). Some of them proposed location centered models (Schilit, 1995). These metamodels are very limited and the applications are very specific. In (Henricksen et al., 2002), the authors proposed models that take care of some contextual entities like place, person, temperature, devices, etc.. These models have been proposed in their own graphical language ORM. As a consequence, it is impossible to exchange model and context information due to

the lack of uniform metamodel.

Recent researches benefit from the standard tools in particular UML/MOF and ontology tools in order to build a uniform infrastructure and to facilitate the use and the exchange of models. As a result, some works focus on defining metamodels to design specific kinds of applications. These works limit the scope of the context information that is considered. However, they take care of the natural diversity of context information. Most of them use an ontology based approach. The relevant works based on such an ontology based approach are summarized in (Strang and Linnhoff-Popien, 2004) and presented in more details in (Matthias et al., 2007). Other recent researches (de Farias et al., 2007b; Sheng and Benatallah, 2005) propose a high abstract level metamodel based on UML/MOF standards. These works, aim at defining and building a uniform infrastructure. However, literature often considers UML as a limited language regarding its capacity to adequately capture contextual information (de Farias et al., 2007b). Indeed, UML is a general purpose modeling language, whose semantics was never intended to accommodate the specific features of ubiquitous applications. Context modeling demands a more specific metamodel. In what follows, we present and discuss the two main approaches for context modeling: UML/MOF and ontology, and we finally give our point of view.

5.2 UML/MOF Approach

Some recent works based on the UML/MOF approach have been proposed to model a mobile context-aware application and mobile services. These applications must react to the constant changes in context within a dynamic environment. Following this approach, the authors in (Sheng and Benatallah, 2005), propose a high abstract level metamodel in which a metamodel named ContextUML is dedicated to context-aware web services. On the one hand, they present a language to model the context for mobile applications; on the other hand, they propose a mechanism to adapt the selected services to the necessary context information available. The proposed context metamodel consists of two main classes the *Context* and the *ContextSource* classes. The first models the context information, and is further specialized by two sub classes *AtomicContext* and *CompositeContext*, whereas the second class, *ContextSource*, models the resource from which the contexts are retrieved. It is also specialized in two sub classes *ContextService* and *ContextServiceCommunity*. This metamodel does not refine contextual information. So, it is hard to identify contextual entities. Another recent work is presented in (de Farias et al., 2007b); It proposes a MOF-based contextual information metamodel, more expressive than ContextUML. In addition the MOF metamodel takes the dynamics of the context into account by adding the notion of time. To best match contextual information with the selected service, they define an environment composed of five views; amongst them the metamodel service view and a context metamodel core view (de Farias et al., 2007b).

5.3 Ontology-based Approach

Modeling context using an ontology-based approach allows describing contexts semantically and provides a vocabulary for representing knowledge about a domain. In addition ontologies are very suitable and applicable to model a specific-domain. Thus a lot of efforts on modeling context based ontologies have been proposed. The majority of them are oriented to describe the context of specific context-aware applications like smart homes, virtual guide tours, smart assistants, etc.

In 2003, a promising emerging context modeling approach, based on ontologies, was proposed for the CoBrA system (Chen et al., 2003). This system provides a set of ontological concepts to characterize entities such as persons, places, intentions or several other kinds of objects within their contexts.

In 2005, in (Gu et al., 2005), the authors propose a metamodel to describe a smart home environment. They define a hierarchical context ontology in which two layers are distinguished: 1) Common upper ontology in which the basic concepts are defined such (e.g.; A person, location, computational entities, activity, etc). 2) The domain-specific ontology in which the details of basic concepts are defined and their properties (eg: person have properties name, role, status, etc.).

Other works, proposing a high level context metamodel, are presented in (Capiello et al., 2005). The authors focus on modeling a general domain which covers a large type of user centered applications. For example applications related to the tourism domain such as virtual traveler's guide, hotel guides, etc.. They define a top ontology that specifies the generic contextual entities and a specific ontology which defines subclasses of the basic element that are defined previously in the top ontology. The define a context top ontology that is composed of many contextual entities such as the user profile, the environment and the channel. The user profile describes the properties associated with the user. The environment consists of location, the ambient condition, the time, etc. Finally the channel describes the element that characterizes

the interaction of the users with the platform used to access services. It includes a device, a network etc.. As an example of specific ontology, a basic concept location specialized in three subclasses: political geography, physical geography and architectural position. The political geography is a street, city, country, state or country. This metamodel uses only two types of relation: aggregation (*part of*) and inheritance (*is a*).

The ontology based approaches are characterized by: (1) Simplicity of use; (2)Flexibility and extensibility by appending a new context elements; (3) Possibility to have a hierarchical context ontology; (4) Expressiveness to describe as many context states as possible. (5) Genericity so as not to be limited to special kinds of context atoms but to allow the definition of generic context classes that can then be specialized by subclasses.

5.4 ODM: A Good Compromise

Although some approaches, e.g. (Sheng and Benatallah, 2005) or (de Farias et al., 2007b) advocate a rather generic knowledge of context-awareness, nowadays no single approach for modeling context provides for all relevant contexts in the necessary detail. The main objective of the ODM is to bridge the gap between traditional software tools for modeling (like UML) and artificial intelligence techniques (Description Logics) for describing ontologies. The principle of ODM is to merge two main domains of research: namely, Model Driven Development and Semantic Web. Basically, ODM allows the description of ontologies using UML (by using a UML profile with existing tools like Rational Rose or Poseidon) and transforming it to OWL/RDF.

5.5 Our Metamodel

As we said above, we are interested in modeling a broad spectrum of user centered context aware mobile applications and thus, have designed a high level context metamodel from which various models can easily be instantiated to describe their own specific context information.

Figure 4 presents our context metamodel which illustrates the most important and generic contextual entities. By contrast to the other works discussed in section 4, our work does not focus on proposing neither, a low level context metamodel for very specific context aware applications for example smart home (Gu et al., 2005). Indeed such a choice does not provide a reusable metamodel. Nor does it attempt to define a high level and very abstract metamodel for the design of mobile applications (de Farias et al., 2007b). In (Sheng and Benatallah, 2005), the authors propose an original approach for modeling context aware mobile applications. Their metamodel for context information is centered on the *Context* entity that represents generic contextual information. This model does not represent the main contextual entities that should be considered in a given mobile context-aware application. In (Broll et al., 2007), the authors present a similar context metamodel and illustrate its use in the development of a mobile application that helps a user finding a restaurant. Also, they introduce the notions of atomic and composite context. However, the entities in the metamodel are in our opinion not described with a sufficient level of detail, and are thus poor semantically.

In this study, we propose a solution to these shortcomings and adopt a UML/ODM/MOF approach to express our vision. Using such an approach it is possible to create a high level metamodel and benefit from all the advantages of the object, model and ontology based approaches. Our context metamodel appears both generic and rich semantically; It identifies and adds the most relevant and generic contextual entities that can be encountered when modeling any mobile and context aware application.

This context metamodel consists of six generic classes and three generic contextual entities. It enriches semantically the *Context* and *Contextual_Entity* classes by adding specific attributes and associations. The *Context* class models context information, it is identified by a name and has two types of relation: the aggregation *includes* and the association *uses*. The first relation expresses that a *Context* is composed of many *Context_Entity*. It illustrates the low level composite context (e.g: a composite context information ”profile” is composed of three atomic contexts: ”name”, ”sex”, ”date of birth”). The second relation *uses* expresses the possibility of using other contextual information to specify a context; this could for instance be past context information encountered by the application. This allows to derive new context information from other contexts (e.g: the user favourite food deduced from the user previous preferences). Furthermore, atomic context information is elementary context information that is acquired from one source (e.g: sensor, database . . .). On the opposite, a composite context is an interpretation of various other context informations. It comprises and relates to various pieces of context information whose combination has a meaning of its own. Composite context information consists of multiple atomic and/or composite context information (Sheng and Benatallah, 2005). The second generic entity of the metamodel is the

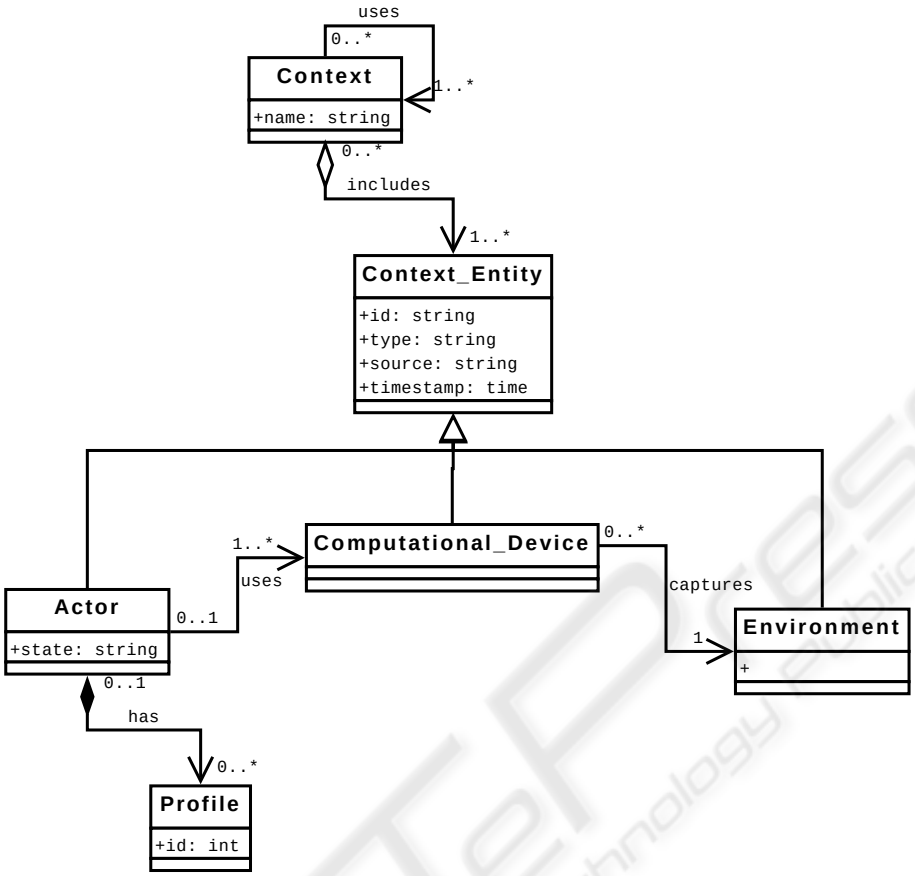

Figure 4: Our context metamodel.

Context_Entity. It is enriched semantically by three attributes:

- The Source. From which the context information is captured. We characterize four main sources: Sensors, actors (users), user applications and derivation from other context information. According to these sources, we classify context informations into three types: (Henricksen et al., 2002):
  - Sensed. Information obtained from hardware sensors (e.g: user location coordinates captured from GPS sensor) or software sensors (eg: the weather information obtained from Internet servers)
  - Deduced. Information obtained from one or more other context information using inference functions or rules. (e.g accordingly to user location coordinates, the user can be located near or far the selected restaurant)
  - Profiled. Information supplied directly by the users (eg: user profiles sush as preferences, constraints) or indirectly through users applications (e.g: software that maintains a history of user preferences)
- Timestamp. To indicate the dynamics of context information

Further, *Context_Entity* is specialized by three main classes: *Actor*, *Environment* and *Computational_Device*.

Actor. It 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”.

Computational_Device. It 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. It 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:

- Spatial context information can be location, city, building ...
- Temporal context information comprises time, date, season ...
- Climate can be temperature, type of weather ...

The last entity is a profile: it is important to mention it here because this entity is capital in any user centered context aware application. In addition profile is strongly attached to the actor and contains the information that describes it. An actor can have both a dynamic and/or a static profile. In fact, 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. He has a pain in his stomach as a constraint.

6 CONCLUSIONS

In this work we have investigated the issue of context information modeling and have proposed an architecture for the development of context-aware mobile applications according to a model driven approach. Context-aware development has been an emergent subject of many research works in ubiquitous computing. However, few of them propose Model Driven Development as an approach for context-aware applications. By the separation of concerns in individual models and by suitable transformation techniques, context can be provided, modelled and adapted independently of business logic and platform details. We have proposed an architecture with three main objective:

- A separation between context information and business logic in individual models;
- The integration of the context model into the business logic using parameterized transformation techniques;
- The mapping of the contextualized business logic model into a web service platform.

REFERENCES

Bézivin, J. (2005). On the unification power of models. In *Software and System Modeling*, volume 4-2, pages 171–188.

Broll, G., Hubmann, H., Prezerakos, G. N., Kapitsaki, G., and Salsano, S. (2007). Modeling context information for realizing simple mobile services. In *Proceedings of the 16th IST Mobile & Wireless Communications Summit*, Budapest, Hungary.

Brown, P. J., Bovey, J. D., and Chen, X. (1997). Context-aware applications: from the laboratory to the marketplace. In *IEEE Personal Communications*, volume 4, pages 58–64.

Capiello, C., Comuzzi, M., Mussi, E., and Pernici, B. (2005). Context management for adaptive information systems. In *Proceedings of the 1st International Workshop on Context for Web Services (CWS'2005)*.

Chen, G. and Kotz, D. (2000). A survey of context-aware mobile computing research. Technical report, Dept. of Computer Science, Dartmouth College.

Chen, H., Finin, T., and Joshi, A. (2003). Ontology for context-aware pervasive computing environments. In *The Knowledge Engineering Review*, volume 18, pages 197–207.

de Farias, C. R. G., Pires, L. F., and van Sinderen, M. (2007a). A case study on the transformation of context-aware domain data onto xml schemas. In Pires, L. F. and Hammoudi, S., editors, *The 3rd International Workshop on Model-Driven Enterprise Information Systems, MDEIS 2007*, pages 63–72, Funchal, Portugal. INSTICC Press.

de Farias, C. R. G., Pires, L. F., and van Sinderen, M. (2007b). A mof metamodel for the development of context-aware mobile applications. In *Proceeding of In he 22nd ACM Symposium on Applied Computing (SAC'07)*.

Dey, A. K. (2001). Understanding and using context. In *Personal and Ubiquitous Computing*, volume 5-1, pages 4–7.

Dey, A. K. and Abowd, G. D. (1999). Towards a better understanding of context and context-awareness. Technical Report git-gvu-99-22,, Institute of Technology, Georgia.

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

Gu, T., Pung, H. K., , and Zhang, D. Q. (2005). A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications*, 28:1–18.

Henricksen, K., Indulska, J., and Rakotonirainy (2002). A. modeling context information in pervasive computing systems. In *Proceedings of 1st International Conference on Pervasive Computing*, Zurich, Switzerland.

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

Matthias, B., Dustdar, S., and Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of ad Hoc and ubiquitous Computing*, 2:263–277.

OMG (2001). Model driven architecture. document. Technical Report ormsc/2001-07-01, Object Management Group (OMG).

OMG (2006). Ontology definition metamodel. Technical report, Object Management Group (OMG).

Pascoe, J. (1998). Adding generic contextual capabilities to wearable computers. In Press, I. C., editor, *2nd International Symposium on Wearable Computers*, pages 92–99, Los Alamitos, California.

Ryan, N., Pascoe, J., and Morse, D. (1997). Enhanced reality fieldwork: the context-aware archaeological assistant. In Gaffney, V., van Leusen, M., and Exxon, S., editors, *Computer Applications in Archaeology 1997*, British Archaeological Reports. Tempus Reparatum.

Schilit, B. and Theimer, M. (1994). Disseminating active map information to mobile hosts. *IEEE Network*, 8:22–32.

Schilit, B. N. (1995). *Context-Aware System Architecture for Mobile Distributed Computing*. PhD thesis, Columbia University.

Sheng, Q. Z. and Benatallah, B. (2005). Contextuml: A uml-based modeling language for model-driven development of context-aware web services development. *International Conference on Mobile Business (ICMB'05)*, 0:206–212.

Strang, T. and Linnhoff-Popien, C. (2004). A context modeling survey. In *Workshop on Advanced Context Modelling, Reasoning and Management*, Nottingham, England.

Vale, S. and Hammoudi, S. (2008). Model driven development of context-aware service oriented architecture. In *Proceedings of the 2008 11th IEEE International Conference on Computational Science and Engineering (CSEWORKSHOPS'08)*, pages 412–418. IEEE Computer Society.

Want, R., Hopper, A., Falc ao, V., and Gibbons, J. (1992). The active badge location system. *ACM Transactions on Information Systems*, 10:91–102.

Weiser, M. (1991). The computer for the 21st century. *Human-computer interaction: toward the year 2000*, 0:933–940.

Winograd, T. (2001). Architectures for context. *Human-Computer Interactions*, 16(2):401–419.