The Development and the Features of the Context-Aware Services

Boudjemaa Boudaa[1], Olivier Camp[2], Slimane Hammoudi[2], Mohammed Amine Chikh[3]
[1]Ibn Khaldoun University
Tiaret, Algeria
[2]MODESTE, ESEO
Angers, France
[3]Abou Bekr Belkaid University
Tlemcen, Algeria
boudjemaa.boudaa@univ-tiaret.dz, {folivier.camp,slimane.hammoudig}@eseo.fr, mea chikh@mail.univ-tlemcen.dz

ABSTRACT: *The development of context-aware applications has been the subject of many research works in pervasive computing. As humans are getting more and more equipped with increasingly powerful mobile computing devices giving them access to online services, the need for personalized and adaptive information services is rapidly growing. Thus, the field of context-aware services has been, during recent years, a field of intense research and has given rise to several approaches. However, in most of the approaches there is a lack of generic methodology for formalizing the development activity for this type of services and, consequently, this activity is very cumbersome and time consuming. Recently, some research works have advocated Model-Driven Development (MDD) as an approach for context-aware services development. In this paper we aim to review these works and their main features.*

Keywords: MDD, Context, Context-Awareness, Context - Aware Service

Received: 12 September 2011, Revised 9 November 2011, Accepted 17 November 2011

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 and have contextual dependencies embedded in them. Such application models are not suited to operate in a pervasive computing environment, which is characterised by richness of context, mobility of users, variety of devices (PDAs, smartphones,. . .) and appearance/ disappearance of resources over time [1].

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 realisation of these pervasive applications which must adapt to different contexts can be developed by Service Oriented Architecture (SOA). Indeed, the loose coupling and interoperability inherent to SOA seems to make this approach the most suitable for context-aware services [2].

Based on the SOA paradigm, various research works for the development of context-aware services have been carried out by

proposing different approaches and methodologies[3]. However, in most of the approaches there is a lack of generic methodology for formalising 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. 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 [4].

In this paper we review the research works presented in the literature and their main features. We consider the approaches based on the OMG (Object Management Group) standards that use the modelling languages UML and the metamodelling language MOF (Meta Object Facility). We also investigate specific approaches based on Domain Specific Languages (DSL). One of the first proposals using MDD for modelling the interaction between context and service for web services is found in ContextUML[5]. This work has influenced several research works and some extended version of ContextUML have been proposed. Thus, our review will discuss the following: The ContextUML approach [5]; Approaches based on ContextUML [6], [7]; Approaches based on UML and MOF [8], [9]. Finally, we will discuss ad hoc approaches based on specific Domain Specific Languages (DSLs) [10], [11].

From the study of the field of context-awareness and the different issues addressed in the literature we have identified seven main issues which will constitute the base of our final comparison between the different approaches (see Section 7).

This review aims to serve as a comprehensive presentation towards the research community regarding the state of the art on recent and representative works in context-aware service engineering using model-driven development.

After the presentation of the theoretical background set for our review in section 2, The four approaches listed above are discussed in sections 3, 4, 5 and 6 respectively. Section 7 discusses, compares and presents the lessons that can be learned from the various reviewed works. Finally, Section 8 concludes and presents future directions of research.

2. Fundamentals

In this section, we aim to define the main concepts referred to in this paper, namely: context, context-aware service and model-driven development.

2.1 Context

What is understood by "*context information*" in contextaware systems and what could be the definition of context have been the subjects of many works. Various definitions of the term are given and summarised in [12].

In early stages, definitions were given for the context of specific applications by enumerating concrete contextual entities. For example the authors in [13] define the context as being information about location, the identity of people in close proximity, physical conditions. In [14], the authors add to this definition the notion of time. Other definitions are extremely broad; the most popular one is given by [15]: "Context is any information that can be used to characterise 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 [16]. In [17], the author approves this definition and claims that it covers all proposed works in context. However he considers it as being 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*". The definition proposed in [18] also presents the context as being hierarchically organised. In this work the authors differentiate between environmental information that determines the behaviour of mobile applications and 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 behaviour or in which an application event occurs and is interesting to the user*". As stated previously, it is difficult to give a complete definition for a context and, in fact, the notion of context is not universal but relative to some situation and application domain [13]. Typically, the context will contain information on the identity of the user, on her activity, on the time at which the context is captured and on the location of the client terminal. Basically, it should answer the following questions "*who*?", "*what*?", "*when*?" and "*where*?" and, ideally, should allow the system to answer one last question : "*why*?".

2.2 Context-Aware Service

New technologies, in particular, wireless communications, together with the increasing use of portable devices (smart phones, personal digital assistants, laptops,...) 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. As he writes in [19], "*appliances should disappear into the background to make the user and his tasks the* central focus rather than *computing devices and technical issues*". 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. This terminology was discussed in [20] 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 [21], the author defines 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 behaviour (at run-time environment) and finally present available and appropriate services to the user. In [15], 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 user's task*".

Context-aware applications should thus be capable of collecting context information and adapting their behaviour to context changes.

In the field of Web services, context is defined by all information about the client of a Web service that may be utilised by the Web service to adjust execution and output to provide the client with a customised and personalised behaviour [22].

Consequently, a Context-Aware Service (CAS for short) is a service which uses context to provide relevant information and/or services to users, where relevance depends on the user's task. A CAS can present relevant information or can be executed or adapted automatically, based on available context information [5]. For example, a service in a tourist information system can display tourist attractions that are close to the user's current location, and if the weather is harsh (for example, if the likelihood of rain is greater than 80% or temperature is above or below some defined thresholds), the service could only suggest indoor activities.

The context of a user may change dynamically, and contextaware services must have the ability to sense context and to react to its changes. Hence, to develop CASs, two important issues need to be considered. The first is the provisioning of context information: CAS developers have to identify what kind of context information will be used and how to collect or derive it. The second issue is the mechanisms that can be used by CASs to adapt their behaviours according to the current context information without explicit user intervention. In other words, the problem to be solved when developing CAS based applications is that of how to use context information to achieve context-awareness of services [5].

In [23], the authors present and describe the basic components that compose context-aware systems in Web services environments (see figure 1). The components in this diagram are either considered as context-aware services and applications, meaning they adapt to context information or as context supporting elements, meaning they provide the applications and services with appropriate context information. Services and applications can thus either invoke other services and applications (in solid black arrowed lines) or retrieve context related information from the context supporting components (in dashed arrowed lines). Moreover, to provide the most relevant context information to a service or an application, the context supporting layer sometimes also has to invoke other context supporting components.

2.3 Model-Driven Development

Model-Driven Development (M DD) is an approach to software development that proposes to use machine-readable models at various levels of abstraction as its main artifacts. The key idea is to automatically transform highly abstract models into more concrete models from which an implementation can be generated in a straightforward way. The MDD approach is supported by the MDA (Model-Driven Architecture) standard of the OMG[24], 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 [25].

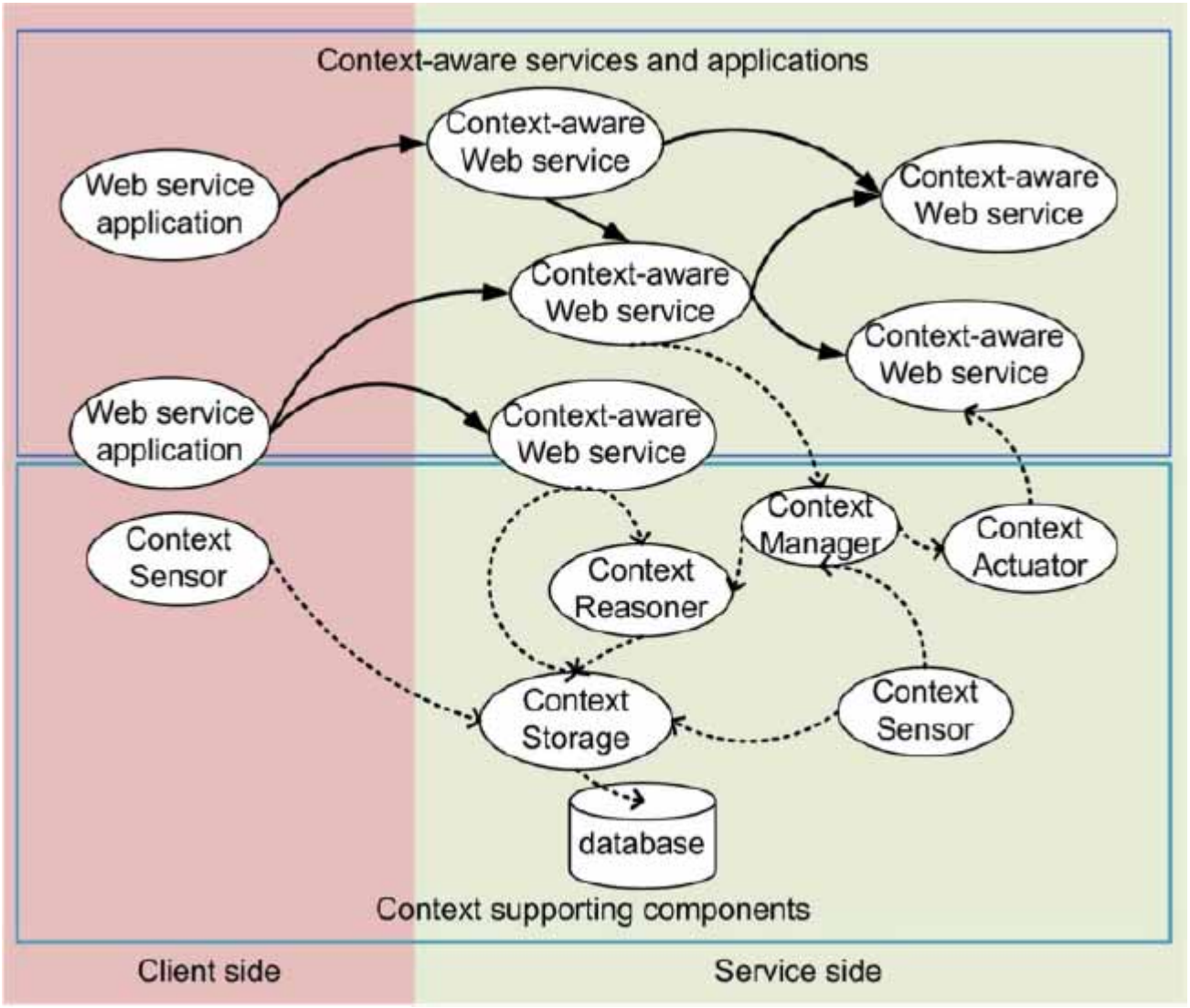

Figure 1. Basic components in a context-aware system in the services environment

Through the clear separation of the business logic expressed in the PIM and of the technical details, contained in the transformation rules that generate the PSM, the MDD approach facilitates the development of pervasive applications for numerous devices. Indeed, a single PIM can accommodate for multiple PSMs required by various devices and platforms of different technologies. Thus, various different embedded pervasive devices, with varying capabilities and requirements, can execute the same application provided the PIM is described and transformations rules are given for each device and platform.

Consequently, another main issue in MDD, is the process of model transformation. Nowadays, it is well recognised that the process of model transformation is one of the most important operations in MDA. The following, largely consensual, definition of model transformation is proposed in [26] "*A Transformation is the automatic generation of a target model from a source model, according to a transformation definition. A transformation definition is a set of transformation rules that together describe how a model in the source language can be transformed into a model in the target language. A transformation rule is a description of how one or more constructs in the source language can be transformed into one or more constructs in the target language*". In the context of the basic four level Metamodelling architecture of MDA, various scenarios of model-to-model transformation have been identified. Figure 2 presents the most common scenario of these transformations, which is compatible with the MOF2.0/QVT standard. Each element presented in this figure plays an important role in MDA. 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 metamodel. 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.

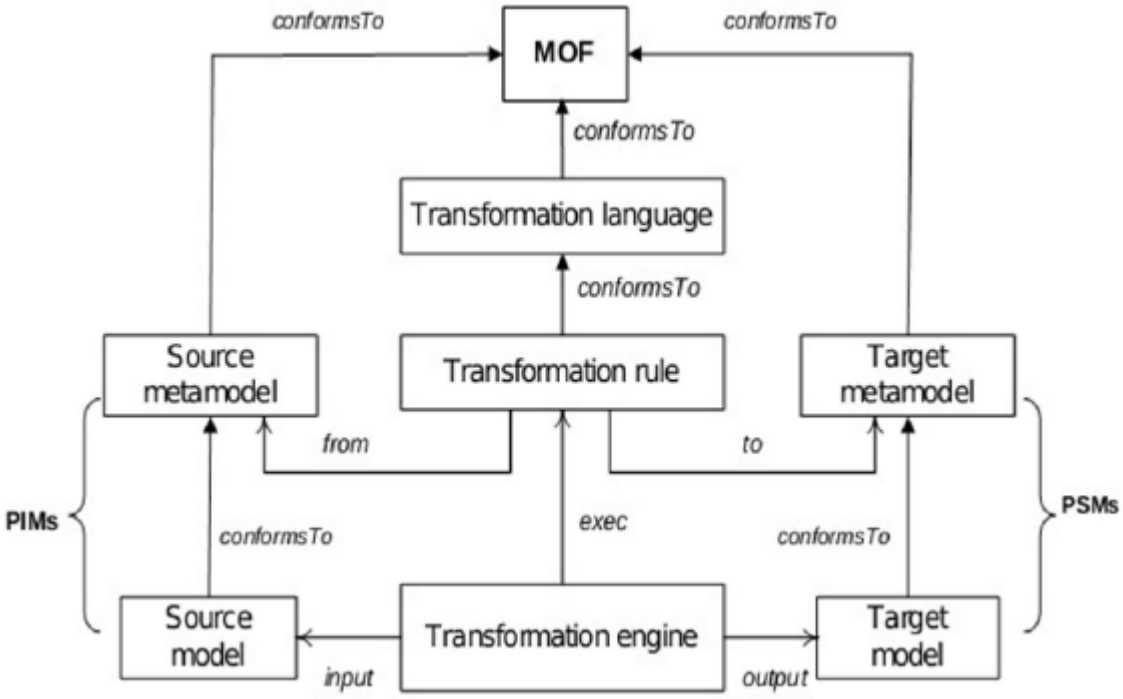

Figure 2. Architecture of the transformation process in MDA

3. Context UML

Using the UML language[27], the work of Sheng and Benatallah [5] is considered one of the first approaches for providing a generic UML metamodel that can be used for contextaware services. In this work, the authors present ContextUML metamodel to design context-aware web services with Model- Driven Development. The main philosophy of ContextUML is that service constructs are in fact context-dependent and thus can be associated with context attributes described by a relevant context model. Within this model, context attributes are populated through the invocation of the corresponding context supporting services[6].

ContextUML includes both a metamodel and a notation language. The metamodel defines the abstract syntax of the language, while the notation defines the concrete syntax used to represent it [5]. Here, we are interested by the ContextUML metamodel which is composed of two separate aspects: modeling of the context and modeling of context-awareness.

3.1 Context Modelling

The *Context* class uses two subclasses to represent atomic contexts and composite contexts. Atomic contexts are lowlevel contexts that do not rely on other contexts and can be provided directly by context sources. Whereas, composite contexts are high-level contexts that may not have direct counterparts on the context provision side. A composite context aggregates multiple contexts, either atomic or composite.

The *ContextSource* class models the resources from which contexts are retrieved. It is specialised by the *ContextService* and *ContextServiceCommunity* subclasses. A ContextService is provided of an autonomous provider, collecting, refining, and disseminating context information. Different ContextServices can provide very different and heterogeneous context information. To solve the problems of heterogeneity, ContextServiceCommunity aggregate multiple context services and provide a unified interface. Another advantage of using communities of context service is the dynamic provisioning of context information. When

the operation of a community is invoked, the community is responsible for selecting the most appropriate context service (at run-time) for providing the requested context information [5].

3.2 Context-Awareness Modelling

A CAMechanism is a class that formalises the mechanisms for context-awareness. Two categories of mechanisms are presented: *ContextBinding* and *ContextTriggering*[5]. Context-awareness mechanisms are assigned to context-aware objects -modelled by instances of *CAObject*- through the MechanismAssignment relation, indicating which objects have what kinds of context-awareness mechanisms. CAObject is a base class for all ContextUML model elements that represent context-aware objects. There are four subtypes of *CAObject: Service, Operation, Message* and *Part*. A context-awareness mechanism can either be assigned to a service, an operation of a service, input/output messages of an operation, or even a particular part of a message.

The *ContextBinding* class models the automatic binding of contexts to context-aware objects (e.g., a service operation's input parameter).

The *ContextTriggering* class models when and how contextual adaptation should occur. A context triggering mechanism contains, both, a set of context constraints (conditions) and a set of actions, and specifies that the set of actions is executed when all the context constraints evaluate to true.

ContextUML has many advantages [5]; it provides context processing mechanisms for the development of CASs and also also offers rich primitives for modelling contexts and their capture. The language also enables the development of context-aware Web Services, supports the modelling of context and offers significant design flexibility to CAS designers. This is achieved through the following major aspects: firstly, the separation between context modelling and context-awareness from service components; several context-awareness mechanisms are abstracted and mechanisms are assigned to relevant service components to achieve context-awareness. This separation facilitates both the development and the maintenance of CASs.

Secondly, the abstraction of context service communities provides a significant flexibility for context provisioning by dynamic binding of context services, and ensures the high quality of context information through quality based selection policies. Communities also make context provisioning more robust. For example, if a selected context service from a community becomes unavailable, another context service can be selected from the same community for sustaining its high availability [28].

Finally, composite contexts improve the modelling power of context information for CAS designers. By applying composite contexts, service designers can model any high-level context attributes that are useful in CASs.

However according to [29], even though ContextUML allows the modelling of derivation rules, it does not include means to model user privacy. Furthermore, since ContextUML, by providing a heavyweight extension of the UML metamodel, modifies the UML metaclasses, it cannot be used by standard UML modelling tools.

Recently, and based on ContextUML, Sheng et al. have presented the ContextServ Platform designed for the rapid development of context-aware Web services[30]. ContextServ adopts model-driven development and uses ContextUML for the specification of CASs. The platform also offers a set of automated tools for generating and deploying executable implementations of context-aware Web services. And in[31], they summarized all their techniques on developing context-aware web services with presenting a concrete prototype contextaware Web application called "*Smart Adelaide Guide*", which help tourists of Adelaide (the capital city of South Australia) to find interesting places, based on their current locations, preferred languages, and weather condition. The Web application relies on a context-aware attractions search Web service that has been developed by using ContextServ Platform.

However, ContextUML 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.

4. Extended ContextUML

To benefit of the work of [5], various approaches have proposed extended and modified versions of ContextUML, we can cite here two interesting works: the work of [6]which discusses an approach based on Model-Driven Design and Aspect Oriented Programming, and the work of [7] that deals with context information in the Simple Mobile Services (SMS) project.

4.1 ContextUML and Aspects

Prezerakos et al., in [6], saw that ContextUML metamodel require several modifications such as (1) minimize the association between the way the services are modeled and the way context is (2) eliminate unnecessary relationships between existing metamodel artifacts and (3) clarify existing metamodel semantics in order to facilitate the translation of the resulting service model into code by a model translation tool. Where they are presented a modified ContextUML model [6]. The differences between the original and the modified ContextUML metamodel fall broadly into two categories: differences in the stereotypes contained in the metamodel and differences in the use of these stereotypes during the creation of a service or context model.

To gain benefits from combination MDD with AOP, this work is one of few works have treated the context with Aspect Oriented Programming (AOP), of which the authors were based on the idea which considers the core service logic and context handling are treated as separate concerns at the modelling level as well as in the resulting source code where AOP encapsulates context-dependent behaviour in discrete code modules (core service functionality and handling of context can be viewed as two separate concerns with context handling crosscutting into the core service functionality). Using the AOP paradigm context information can be handled through aspects that interrupt the main service execution, in order to achieve service adaptation to context [3].

In the design phase of [6], Prezerakos et al. have described a modeling process which consists of the following steps: (a) Discovery of relevant Web Services and their translation to UML diagrams (b) Design of a context and service logic model based on these UML diagrams and the ContextUML metamodel, (c) Association of context model elements with respective context sources, (d) Association of service logic elements with respective context model elements and (e) manual assembly of the discovered Web Services into a composite workflow using a UML activity diagram.

In the coding phase, service and context UML models, resulting of the above phase, are exported as XMI files and fed into a model translation tool which, driven from the stereotypes of the ContextUML model, converts them into Java source. The usage of an AOP engine is assumed as the mechanism that enables the extension of existing services behavior. Context binding information provided in the UML models is used to create pointcuts and related advices, as well as to create the binding between them.

The effective use of MDD techniques, in this approach, allows service developers to handle business logic and contextdependent service behaviour as two separate concerns within the same model, enabling the modification of the contextdependent behaviour without severely affecting the main functionality of the service. The same separation of concerns can be achieved at the source code level and consequently during service operation by encapsulating context-dependent functionality into separate aspects in a high level programming language.

4.2 Modelling Context Information for Realising Simple Mobile Services

In [7], the authors present a UML-based context model for the Simple Mobile Services (SMS) project. The SMS system focuses on modelling, managing and providing context information in order to facilitate the authoring, provisioning and usage of context-aware mobile services.

To integrate context information into the authoring of services and the modelling of their context-aware behaviour, a context model for SMS has been developed. This model, inspired by ContextUML approach[5], includes three different levels of abstraction (metamodel, model and instance). The resulting UML model can be used to derive context models using other languages, e.g. XML-related standards like 3GPP. These levels offer different levels of abstraction and can be used as a basis for transforming the resulting model to various implementation languages.

The Metamodel level of the SMS context model defines generic entities for the modelling of context, its structure, properties and relations with a high level of abstraction and independently from concrete context information (e.g. location, device,. . .). At this level, the SMS approach uses ContextUML's Context, AtomicContext and CompositeContext classes, but simplifies the modelling of context sources and adds its own class for describing quality of context parameters[7].

However, This proposed context metamodel focus on a specific application domain like "*find a restaurant in a city*", design of a "*smart home*", etc...

5. UML and MOF Approaches

Beside ContextUML, several other approaches have beenproposed using the OMG standards UML and MOF. Two of the most important of these approaches are presented hereafter. The first one, described in [8], proposes a MOF metamodel for the development of context-aware mobile applications. The second approach presented in [9]proposes a methodology and advocates in favour of a complete separation between the functionalities of the web application and of the context adaptation at all development phases (analysis, design, implementation).

5.1 A MOF Metamodel for the Development of Context-Aware Mobile Applications

The Meta Object Facility (MOF) specification [32] is the foundation of the OMG metamodelling strategy and can be used to design the abstract syntax of any metamodel. MOF's abstract syntax is represented using MOF itself and uses a UML class diagram as its concrete syntax. So, we can assume that any MOF metamodel can be represented with UML tools, since an instance of a MOF model is also a UML class diagram.

In [8], the authors present a MOF-based contextual information metamodel for the development of context-aware applications. The metamodel is structured according to the following five views : core, service, subscription, contextaware service and quality views where each view is represented by a MOF package. The two main views are the core and service views. The core view is used for representing the core concepts of any context. Whereas the service view is for modelling the context-aware application as a collection of context-aware services in order to ensure the interaction and interoperability between the context-aware applications and the service platform (service-oriented architecture, in this case) by using the mechanisms which should ensure loose coupling between application and platform, provide a high degree of flexibility and allow modifications of the context without implications on the application, be adaptable and accept different context models sharing the same metamodel and, finally, be independent from any specific context platform.

5.2 Model-Driven Development of Composite Context-Aware Web Applications

The approach cited in [9] proposes an architecture for the context adaptation of web applications consisting of web services and a model-driven methodology for the development of such context-aware composite applications. This methodology adopts a Model-Driven Engineering approach that uses UML diagrams [27] during the design phase. It is shown that the concrete UML model is sufficient to automatically generate a functional web service based web application through an adequate transformation process.

At the modelling level, the design is, to a great extent, kept independent from specific platform implementations and sufficiently flexible to allow the introduction of different code specific mappings. During both the application's development and execution phases the service logic and the context adaptation are kept independent, thus providing flexibility and facilitating the maintenance of the application.

The modelling exploits a number of predefined profiles (a Web service profile, a context meta-model and a presentation profile), whereas the target implementation is based on an architecture that performs context adaptation of web services based on interception of Simple Object Access Protocol (SOAP) messages [3]. Furthermore, it is shown that the context adaptation process is entirely transparent to the core web application.

6. Domain Specific Language based Approach

A number of works, among which [10] and [11], have also proposed Domain Specific Languages (DSLs) for the modelling of context-aware services. A DSL is a programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain. As compared to the approaches presented in the previous sections, the solutions presented here are, thus, not based on UML but rather on a dedicated language specifically designed to address the problem of context-awareness.

6.1 Model-Driven Development of Context-Aware Services

In [10], the authors present a model-driven design trajectory for context-aware and mobile services. In this work, a number of concepts such as platform independence, abstract platforms, context-awareness and service orientation play an important role. The authors present the design trajectory by discussing the necessary levels of models, the choice of modelling languages and the definition of platforms and transformations.

Their design process is composed of two main phases: the preparation phase and the service creation phase. In the preparation phase, experts identify the required levels of models, their abstract platforms and the modelling languages to be used. In addition, this phase also defines transformations between related levels of models. The results of the preparation phase are used in the service creation phase. The latter entails the creation of models of a specific service using specific modelling languages and abstract platforms and applying (manually and automatically) transformations to models. Ultimately, the service creation phase leads to a realisation of the context-aware service that satisfies the user requirements.

Almeida et al. decompose the context-aware service according to an architecture which consists of context sources, which are able to sense context and represent it as context information in the scope of the system. The service provided by context sources is used by a coordination component, which requests actions to be executed by action providers depending on situations that can be inferred from context information.

In this architecture, the user components and the coordination component exhibit service-specific behaviour, and are called service components. In contrast, context sources and action providers are general-purpose and, therefore, can be reused in several different context-aware services. The service provided by context sources and action providers to the coordination component is registered in a service trader. This allows the coordination component to select context sources and action providers dynamically according to service offers that are registered in the service trader.

The authors use the A-MUSE Platform which provides an abstraction of middleware and service discovery platforms and includes context and action services, and, in order to facilitate the transformation from the service specification level (high level of abstraction) to the service realisation level (low level), introduce an intermediate level of models.

These three levels of abstraction present the main advantage of this approach model-driven development.

6.2 Context-Dependent Role Model

Another approach of this category is presented by Vallejos et al. in [11]. They propose a role-based object-oriented programming model, called the context-dependent role (CDR) model, to facilitate the development of context-dependent adaptations in mobile distributed systems. In this model: (1) roles represent the different behavioural adaptations a software application can dynamically adopt according to the context, (2) an application autonomously decides of an appropriate role based on the context of all the participants, and (3) an adaptation is strictly delimited by the scope of an interaction.

In the CDR model, an actor encapsulates a delegation hierarchy composed of a default behaviour and its different contextdependent adaptations, all of them represented as roles; actors respond to messages by first selecting the appropriate role and then executing the corresponding method in the adaptation object of that role; the role required for the execution of a message is autonomously selected by the actor that receives the message, using the context-dependent role selector and based on the context of both the sender and receiver of the message; adaptations have a delimited scope of action which is defined by the execution of a message; and a context reference enables the message sender to be aware of the part of the context exposed to the message receiver.

7. Comparison and Lessons Learned

Based on [3], [33], table I summarises the most representative approaches in Model-Driven Development for contextaware services according to: 1) the modelling language used, 2) the degree of decoupling between the business logic and context management, 3) the compliance with the MDA standard (PIM, PSM, transformation process,...) 4) the time of adaptation to context information (design-time or run-time) 5) the side on which context adaptation occurs, 6) the fact whether the approach deals with the security and privacy aspects and finally, 7) whether there is a platform or framework implementing the approach.

We observe that the majority of the discussed approaches succeed in decoupling the context management from the business logic to a certain degree. For context modelling, these approaches use modeling languages that are either standard and generic (e.g. UML) or specific (in the case of DSL). Other approaches cited in [3], [33] use code-based and messageinterception techniques for the representation of context. These techniques are out of the scope of this paper and are not discussed here. Unfortunately, even though we believe security and privacy, compliance with the MDA standard and dynamic contextual

adaptation at run-time are major issues for contextaware services[1], they are either partially or totally ignored by these approaches. These limitations represent interesting research topics in the future for the development of CASs, as follows:

• MDA compliance : MDA is the most recent software engineering approach, initiative of OMG as a Model-Driven Engineering standard. Adopting MDA-based modeling and development approach for context-aware services responds to many software engineering requirements such as, interoperability, genericity and reusability.

• Dynamic contextual adaptation: the context with its frequent changing nature requires the rapid and dynamic adaptation of CAS at runtime without interrupting service availability. Such dynamic adaptation is only dealt with by few works and often considered at the code level using, for instance, aspect oriented programming. We believe dynamic adaptation of CASs should be considered at a much more abstract level using models. Subjects such as Aspect Oriented Modelling or Models@Runtime, developed today by the MDE community, seem to us very promising fields.

• Security and privacy : Privacy is the claim of individuals, groups and institutions to determine for themselves, when, how and to what extent information about them is communicated to others [34]. All CAS developments aiming to achieve privacy, should implement the existing laws and regulations to protect the individual's privacy, namely: Data must be kept secret; Data should be accessible by the subject it refers to; Data should only be collected for a clear purpose; Adequate security safeguards should be put in place. Security and privacy are not our research focus and we will thus not be concerned by this point in our work.

Different implementation languages or underlying frameworks and middleware are adopted in each approach, but in most cases a certain degree of independence is maintained for the service development (i.e. the adaptation takes place without affecting the underlying middleware technology). The adaptation may take place on either side (client or server), whereas there are also cases where both are seen as equivalent actors.

	Context modelling	Decoupling context from usiness logic	Compliance with MDA standard	Time of contextual adaptation	Side of contextual adaptation	Privacy and security	Implementation (platform / framework)
ContextUML [5]	UML-based	Yes	No	@design-time	Client-Side	No	Web services
AOP [6]	UML-based	Yes	No	@run-time	Client-Side	No	Web services
SMS Project [7]	UML-based	Yes	Partial	@design-time	Server-Side	No	Various
MOF metamodel [8]	UML-based	Yes	Partial	@design-time	Server-Side	No	Infraware platform
Composite context-aware[9]	UML-based	Yes	Partial	@run-time	Client/Server-Side	No	Web services
A-MUSE Service Platform [10]	ECA-DSL	Yes	Partial	@design-time	Server-Side	No	A-MUSE Service Platform(Web services,BPEL)
CDR model [11]	Impl.-based Partial	Partial	No	@run-time	Client/Server-Side	Yes	AmbientTalk framework

Table 1. A Synthesis of Development Context-Aware Services Approaches

8. Conclusion and Future Works

In this paper, we have presented a review on approaches and methodologies for Model-Driven Development of Context- Aware Services. We have described and classified the different solutions presented in the literature. One of the characteristics shared by all these approaches is the clear separation between the business logic of the application and context management. Such separation of concerns allows for the same context to be reused throughout different business logic and inversely for different contexts to be adapted to the same application.Designing context-aware services is a complicated task. By raising the level of abstraction using models and meta models, and by using the main techniques of MDA such as automatic (or semi-automatic) model transformations and code generation, the designer can facilitate this cumbersome task. Our future directions of work aim

[1]There are other important issues concerning CASs discussed in [3], such as, for example, historical data support, but in this work we only focus on aspects related to modelling and adaptation.

at providing an MDAbased approach to develop CASs. This approach should leverage the benefits of Model Driven Architecture and Aspect- Oriented Modeling (AOM) [35] while following a methodology providing the complete independence between the business logic and context management mechanisms. Our approach should also take into account context management both at design time and execution time.

We propose to investigate the following techniques in our future research:

• A context metamodel that comprises the main entities used in the state of the art of user centered mobile context aware applications. We feel that ODM (Ontology Definition Metamodel) which allows to create ontologies using UML is an interesting compromise between the very structured UML approach and the flexibility and richness of representation of ontologies.

• Aspect Oriented Modeling [36] to allow the modeling, at design time, of both possible evolutions of the business logic and contextual situations which ensure service adaptability.

• Models@Runtime [37] to allow the implementation of either the evolution of a service's business logic or, most important in our case, of its adaptation at runtime.

• Model weaving and composition at runtime [38], to allow dynamic adaptability.

By following this roadmap we aim to improve the weaknesses of current approaches of MDD for CAS both from a conceptual and an implementation point of view.

Referenses

[1] Vukovic, M., Robinson, P. (2004). Adaptive, planning based, web service composition for context awareness, Advances in Pervasive Computing, 176, 247–2524.

[2] Grassi, V., Sindico, A. (2007). Towards model driven design of service-based context-aware applications, *In*: Proceedings of the International Workshop on Engineering of Software Services for Pervasive Environments, ESSPE 2007 (A. L. Wolf, ed.), (Dubrovnik, Croatia), p. 69–74.

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

[4] Vale, S.,Hammoudi, S. (2005). Context-aware model driven development by parameterized transformation, *In*: Proceedings of the 1st International Workshop on the Model Driven Interoperability for Sustainable Information Systems (MDISIS'08) held in cunjunction with CAiSE'08, (Montpellier, France).

[5] Sheng, Q. Z., Benatallah, B. (2005). Contextuml: A uml-based modeling language for model-driven development of context-aware web services, *In*: Proceedings of 2005 International Conference on Mobile Business (ICMB 2005), (Sydney, Australia), p. 206–212, 11-135, July.

[6] Prezerakos, G. N.., Tselikas, N. D., Cortese, G. (2007). Model-driven composition of context-aware web services using contextuml and aspects, *In*: Proceedings of 2007 IEEE International Conference on Web Services (ICWS 2007), (Salt Lake City, Utah, USA), p. 320–329, IEEE Computer Society.

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

[8] de Farias, C. R. G., Leite, M. M., Calvi, C. Z., Pessoa, R. M., Filho, J. G. P. (2007). A MOF metamodel for the development of context-aware mobile applications. *In*: SAC '07: Proceedings of the 2007 ACM symposium on Applied computing, (New York, NY, USA), p. 947–952, ACM.

[9] Kapitsaki, G. M., Prezerakos, G. N., Tselikas, N. D. (2009). Venieris.Model-driven development of composite context-aware web applications, Information & Software Technology, 51(8) 1244– 1260.

[10] Almeida, J. P. A., Iacob, M.E., Jonkers, H., Quartel, D. A. C. (2006)..Model-driven development of context-aware services, *In*: Proceedings of Distributed Applications and Interoperable Systems, 6th IFIP WG 6.1 International Conference, DAIS 2006 (F. Eliassen and A. Montresor, eds.), V. 4025 of Lecture Notes in Computer Science, (Bologna, Italy), p. 213–227, Springer.

[11] Vallejos, J., Ebraert, P., Desmet, B., Cutsem, T. V., Mostinckx, S., Costanza, P. (2007). The context-dependent role model, in Proceedings of the 7th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS 2007), Lecture Notes in Computer Science, (Paphos, Cyprus), Springer-Verlag.

[12] Bazire, M., Br´ezillon, P. (2005). Understanding context before using it, in Modeling and Using Context, *In*: 5th International and Interdisciplinary Conference, CONTEXT 2005 (A. K. Dey, B. N. Kokinov, D. B. Leake, and R. M. Turner, eds.), V. 3554 of Lecture Notes in Computer Science, p. 29–40, Springer.

[13] Brown, P. J., Bovey, J. D., Chen, X. (1997). Context-aware applications: From the laboratory to the marketplace, IEEE Personal Communications, 4(5) 58–64.

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

[15] Dey, A. K., Abowd, G. D. (1999). Towards a better understanding of context and context-awareness, Tech. Rep. git-gvu-99-22, Institute of Technology, Georgia.

[16] Dey, A. K., Abowd, G. D., Salber, D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications, Human-Computer Interaction Journal, 16 (2) 97–166.

[17] Winograd, T. (2001). Architectures for context, Human-Computer Interactions, 16 (2) 401–419.

[18] Chen, G., Kotz, D. (2000). A survey of context-aware mobile computing research, tech. rep., Dept. of Computer Science, Dartmouth College.

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

[20] Schilit, B., Theimer, M. (1994). Disseminating active map information to mobile hosts, IEEE Network, 8, 22–32.

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

[22] Keidl, M., Kemper, A. (2004). Towards context-aware adaptable web services, *In*: Proceedings of the 13th international conference on World Wide Web - Alternate Track Papers & Posters, WWW 2004 (S. I. Feldman, M. Uretsky, M. Najork, and C. E. Wills, eds.), (New York, NY, USA), ACM.

[23] Truong, H. L., Dustdar, S. (2009). A survey on context-aware web service systems, International Journal of Web Information Systems, 5(1) 5–31.

[24] http://www.omg.org/mda/.

[25] Ayed, D., Berbers, Y. (2006). Uml profile for the design of a platformindependent context-aware applications,in Proceedings of the 1st workshop on MOdel Driven Development for Middleware, MODDM 2006 (I. Gorton, L. Zhu, Y. Liu, and S. Chen, eds.), vol. 183 of ACM International Conference Proceeding Series, (Melbourne, Australia), p. 1–5, ACM.

[26] Kleppe, A.,Warmer, J., Bast, W. (2003). MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley.

[27] OMG. (2010). Unified modeling language (omg uml) infrastructure, 2 (3),http://www.omg.org/spec/uml/2.3.

[28] Maamar, Z., Sheng, Q. Z., Tata, S., Benslimane, D., Sellami, M. (2009). Towards an approach to sustain web services high-availability using communities of web services, International Journal of Web Information Systems, 5, 32–55.

[29] Simons, C. (2007). Cmp: A uml context modeling profile for mobile distributed systems, *In*: Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS '07), (Hawaii), p. 289, IEEE Computer Society.

[30] Sheng, Q. Z., Pohlenz, S., Yu, J., Wong, H. S., Ngu, A. H. H., Maamar, Z. (2009). Contextserv: A platform for rapid and flexible development of context-aware web services, *In*: Proceedings of 31st International Conference on Software Engineering, ICSE 2009, (Vancouver, Canada), p. 619–622, IEEE.

[31] Sheng, Q. Z., Yu, J., Segev, A., Liao, K. (2010). Techniques on developing context-aware web services, IJWIS, 6 (3), 85–202

[32] OMG. (2006). Meta object facility (mof) core specification, omg available specification, version 2.0, http://www.omg.org/spec/mof/2.0/pdf/.

[33] Georgi, G. N. P., Kapitsaki, M., Tselikas, N. D. (2010). Context-Aware Web Service Development: Methodologies and Approaches Chapter in book . Enabling Context-Aware Web Services: Methods, Architectures, and Technologies. Chapman & Hall/CRC, 1st ed.

[34] Westin, A. (1970). Privacy and freedom. New York: Atheneum.

[35] Abeywickrama, D. B., Ramakrishnan, S. (2012). Context-aware services engineering: Models, transformations, and verification, ACM Trans. Internet Technol., 11, 10,1–10:28.

[36] Carton, A., Clarke, S., Senart, A., Cahill, V. (2007). Aspect-oriented modeldriven development for mobile context-aware computing, in SEPCASE '07: *In*: Proceedings of the 1st International Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments, (Washington, DC, USA), p. 5, IEEE Computer Society.

[37] J´ez´equel, J. M. (2010). Ing´enierie Dirig´e par les Mod'eles : du design-time au runtime,G´enie Logiciel - Ing´eni´erie dirig´ee par les mod'eles, 93.

[38] Morin, B., Barais, O., Jezequel, J. M., Fleurey, F., Solberg, A. (2009). Models@ run.time to support dynamic adaptation, Computer, 42, 44–51.