An Aspect Based Pattern for Context-Awareness of Services*

Hatim Hafiddi, Hicham Baidouri, Mahmoud Nassar and Abdelaziz Kriouile,

IMS Team, SIME Lab., ENSIAS, Rabat, Morocco

Summary
Today, service in oriented applications need to be enhanced to sense and react to user's context in order to provide a better user experience. To meet this requirement, Context-Aware Services (CAS) have emerged as an underling design and development paradigm for the development of context-aware applications. The fundamental challenges for such applications development are context management and service adaptation to the user's context. To cope with such requirements, we provide in this paper a context, context provider and CAS specifications and metamodels, and a tool to enhance a core service, in service oriented applications, to be context-aware. This enhancement is satisfied across a Aspect based pattern which, inspired by the Aspect Paradigm (AP) concepts, considers the service's adaptations as aspects.

Key words:
Context, Context-Awareness, SOA, Context-Aware Service, Aspect Paradigm.

1. Introduction

Today, service oriented applications need to be enhanced to sense and react to user's context in order to provide a better user experience. To meet this requirement, Context-Aware Services (CAS) [2] have emerged as an underling design and development paradigm for the development of context-aware applications. A CAS provides users with a customized and personalized behavior depending on their contexts. For example, a *Restaurants Searching* service gives tourists suggestions depending on their locations, preferences and even the used device capabilities. Generally, this kind of information is called context. CAS driven development of service oriented applications enables them to sense and react to the changes observed in their environment. This capability is particularly critical in ubiquitous environments, where context is the central element of mobile systems [18].

The ambiguity of the context concept and the multiplicity of the execution contexts of services make CASs hard to build. Moreover, traditional approaches for CASs development produce services, generally both platform and domain dependent, which are able to function only in preset situations. The business logics of such services are tightly coupled with both of context management and adaptation logics. Consequently, the result of such approaches is complex services whose rate of evolution and reuse is much reduced.

CASs development can benefit from Aspect Paradigm (AP) and Model Driven Engineering (MDE). AP [6] allows the modification of applications with so-called aspects. Aspects are modular units of functionality used across the application code and woven at so-called pointcuts that allow to transparently extending application functionalities. In our approach, adaptations of a given service to its use contexts are seen as aspects. MDE is a model centric approach for software development, in which models are used to drive software development life cycle. In our approach, we provide context, context provider and CAS metamodels that will guide the design of context-awareness models. We have presented in [1] a design process for CASs and we focus in this paper to present the aspect based pattern to enable efficient CASs development.

The outline of this paper is as follows. We present in next section a scenario that concerns a context-ware E-tourism system and highlight the context-awareness challenges. In Sect. 3, we describe our context specification and metamodel. We present in Sect. 4 the context provider specification and metamodel. Sect. 5 presents our CAS specification and metamodel. We show, in Sect. 6, how can AP be applied to fulfill CASs adaptation. Sect. 7 briefly compares related work. Finally, we conclude the paper with plans for future work.

2. E-tourism Motivating Scenario: Restaurants Searching Service

The following scenario illustrates the potential benefits of context-awareness for E-tourism systems:

"*Let's take a tourist who wants to taste the local gastronomy of a city, which he is visiting, so he connects himself via his mobile device (e.g., PDA, iPhone, BlackBerry, etc.) to a context-aware E-tourism system in order to obtain a list of suitable restaurants. He subscribes to the system and launches his request. The service returns an adequate list of restaurants (restaurants availability is taken into consideration), close to his site (taking into consideration the GPS localization),*

* This work is supported by the FNSRSDT under the CSPT-ICTESAD project

Manuscript received January 5, 2012
Manuscript revised January 20, 2012

described in his language (the system will consider the user's language) and taking account his preferences (e.g., food preferences, restaurants prices, etc). Also, let's note that such a system resorts, if necessary, to a results pagination mechanism to avoid its blocking (considering the device capacities, the RAM in this case) and if ever it detects any change in tourist's context (e.g., weak battery or switching of connection mode from a high mode to a low one), it will automatically adapt its behavior (e.g., returned restaurants information will not include photos) in purpose of optimization (i.e., reducing latency and saving battery)."

The development of such E-tourism systems, in particular, and context-aware systems, in general, imply several challenges. First, context definition (i.e., which context information are relevant for the adaptation of the system) and acquisition (i.e., sensing context from the environment) is not an evident process. Second, context-aware systems should autonomously detect relevant changes in the context and react to these changes by either adapting or invoking services. Finally, the adaptation process must be based on mechanisms in accordance with best practices of software engineering in order to produce well designed CASs.

3. Context

Context is the information that characterizes the interactions between humans, applications, and the environment [19]. Context information is domain specific, as a type of information might be considered as context information in one domain but not in another one. Several context definitions were proposed in the literature (e.g., [25], [27], etc.) serving various domains, however the context definition given by Dey and Abowd remains the most referred. In fact, these authors have defined context as "*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*" [29]. As given in [3], we consider context parameters as any additional information that can be used to improve the behavior of a service in a situation. Without such information, the service should be operable as normal but with context information, it is arguable that the service can operate better or more appropriately [10].

Rather than giving context formalization, case of figure for several researches, sometimes domain specific and sometimes generic but not very extensible, we choose to propose a metamodel which is, at the same time, generic and abstract. So, in our specification (see Fig. 1) a context is a set of parameters (e.g., language, localization, battery, connection mode, etc.) and entities (e.g., user, device, etc.) that can be structured on sub contexts. Sub contexts can also be recursively decomposed into categories. Context may be constituted of simple parameters (e.g., language), derived parameters (i.e., computed from other parameters; for example a distance parameter can be computed from two GPS positions) and complex parameters (e.g., GPS) which have representations (e.g., DMS (Degrees, Minutes, and Seconds) and DD (Decimal, Degrees) representation for the localization parameter).

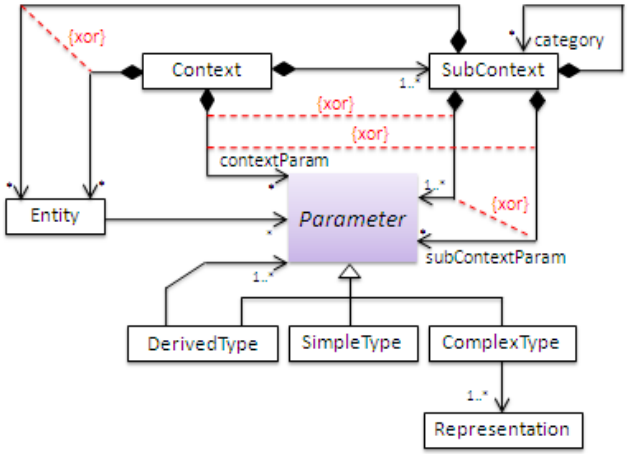

Fig. 1 Core context metamodel.

To illustrate our metamodel, let's project it on the case of figure of the E-tourism system presented in the second section. The context for this system is composed mainly of the following sub contexts (see Fig. 2):

- *DeviceSubContext*: it contains parameters that describe the entity *Device*. It breaks up into two categories which are the software category (e.g., operating system, navigator type, supported type of data, etc.) and the hardware category (e.g., processor type, battery level, memory size, etc);
- *UserSubContext*: it is a sub context that contains parameters describing the entity *User* (e.g., preferences, localization, profile, etc);
- *EnvironmentSubContext*: this sub context contains the *Environment* parameters (e.g., time, weather, etc);
- *ServiceSubContext*: it contains parameters that characterize a *Service* (e.g., price, availability, response rate, etc).

4. Context Providers

The role of context providers is to gather context information from different sources such as sensors, web services, databases, etc. the process of collecting context

depends on context parameters nature and its sources. For instance, the user profile information is explicitly provided

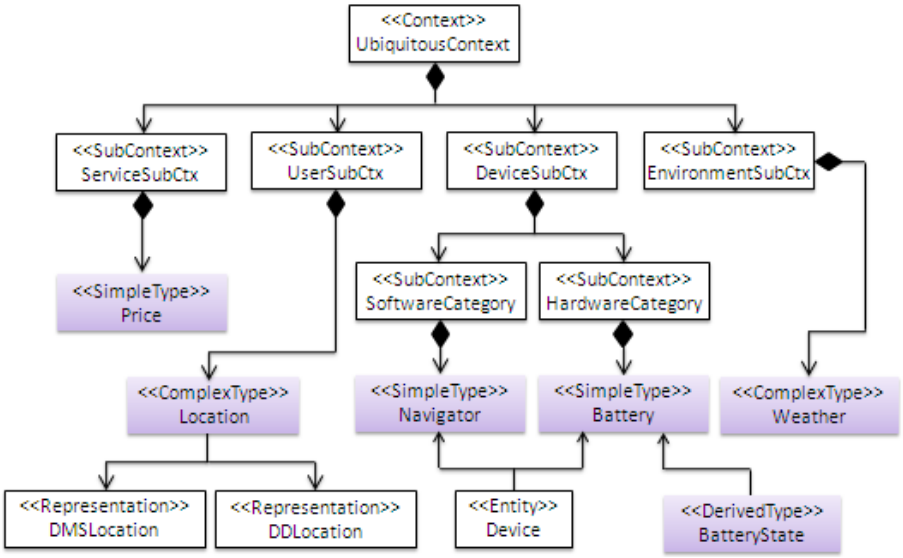

Fig. 2 Succinct context model for the E-tourism scenario.

by the user and so they are characterized by an infrequent change. However, context parameters collected from sensors are subject to frequent changes. Its collection requires interaction with distributed and heterogeneous software or hardware sensors. Also, some context parameters may aggregate or use different context providers to be gathered.

To abstract Context-Aware Applications developers from sensors and sensed data variety and complexity, we provide a context provider specification that abstracts application development stakeholders from sensors API details.

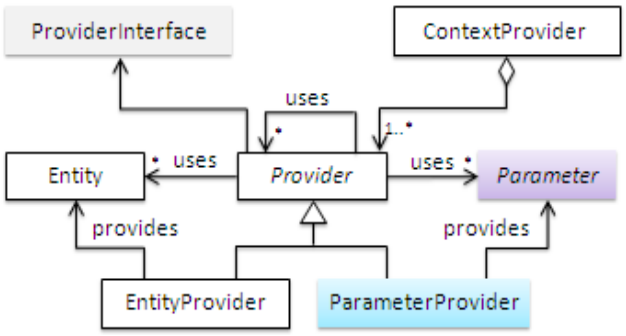

Fig. 3 Core context provider metamodel.

In our specification, as illustrated in figure 3, a context provider (i.e., collector of a given service execution context) aggregates a set of parameters providers (e.g., LocationProvider, WethearProvider, etc.) and entities providers (e.g., UserProvider, DeviceProvider, etc.). Both of entities providers and parameters providers dispose of an interface that specify whether the provider is remote (e.g., a web service that provides weather) or local (e.g., GPS sensor in a mobile device) and what mode of requests is supported (i.e., query-based or notification-based). A provider may use some providers, parameters or entities to get or derive context information. For example, a weather provider uses the localization provider to get the weather information.

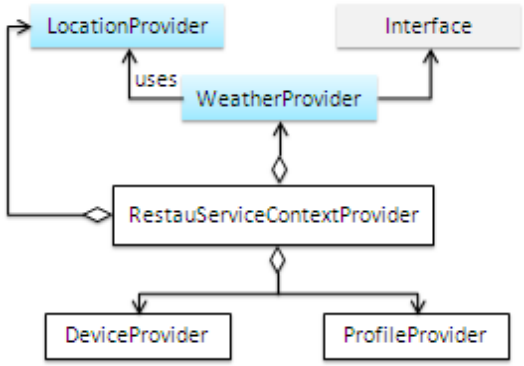

Fig. 4 Succinct context provider model for the Restaurant Service.

Figure 4 gives a succinct context provider model in the case of figure of the Restaurants Searching Service. This provider is composed of the device provider that gathers device context information (e.g., RAM, Battery Level, etc.), the profile provider that provides the profile context information (e.g., age, preferences), the location provider and the weather provider. This last uses the location provider to get the weather information and possesses an interface that specifies how to interact with it.

5. Context-Aware Services Layer

One of the first uses of the term context-aware appeared in 1994 [17]. A service is context-aware if it provides customized and personalized behavior to users depending on their contexts [29]. In Service Oriented Computing (SOC), a service is defined as self-describing and platform-agnostic computational element that supports rapid, low-cost and easy composition of loosely coupled and distributed software applications [24].

To be context-aware, a service must be able to adapt dynamically its behavior to its several execution (i.e., use) contexts. In other words, the service (i.e., core service) must possess mechanisms in purpose to exploit only relevant information of the execution context and adapt dynamically its behavior. Henceforth, this appropriate context information related to a specific execution situation forms what we call the ContextView of the service, and the result of the service adaptation to this ContextView forms the ContextViewService (see Fig .5).

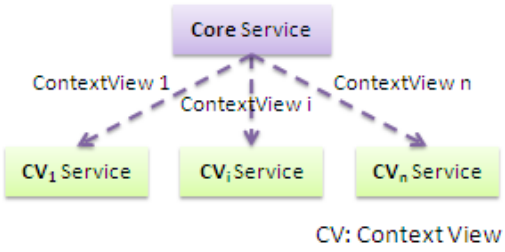

Fig. 5 Core service adaptation to its various ContextViews.

Fig. 6 illustrates our *CAS* metamodel. Accordingly, CAS is seen as a specific service with a number of *ContextView*s. For each one, we associate an adaptation strategy (i.e., *CVSAdaptationStrategy*) which indicates when (i.e., *AdaptationCondition:* classical condition expressed on *ContextView* parameters) and how (i.e., *AdaptationRule:* defines the place in the service where the dynamic adaptations will be realized) a set of ordered adaptations (i.e., *Adaptation*) must be applied, on the core service, in order to provide the expected behavior regarding the current execution context. The adaptation result forms the *ContextViewService*. So, for a given service, the set of its *ContextViewService*s (respectively *CVSAdaptationStrategies*) forms the CAS (respectively *CASAdaptationStrategy*).

For instance, for the E-tourism motivating scenario (c.f. Sect. 2), battery level and connectivity mode present one of the *Restaurants Searching* service *ContextViews*, that will provoke service adaptation, by reducing the amount of data returned (i.e., *Adaptation*) whenever this level is lower than 20% or the connectivity is changed from a high connectivity to a low one (i.e., *AdaptationCondition*). Fig. 7 presents a succinct *CAS* model in the case of the *Restaurants Searching* service.

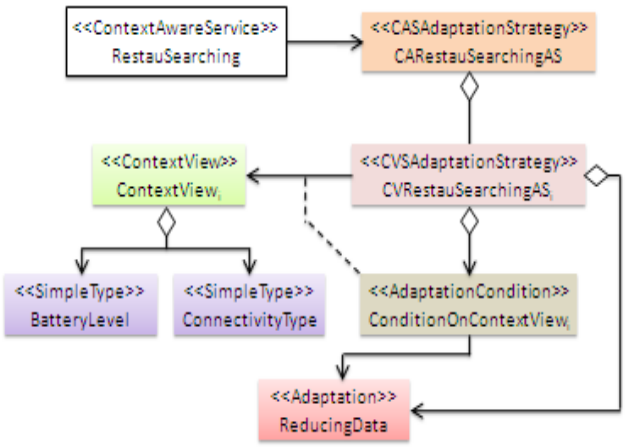

Fig. 7 Succinct CAS model for the restaurants Searching Service.

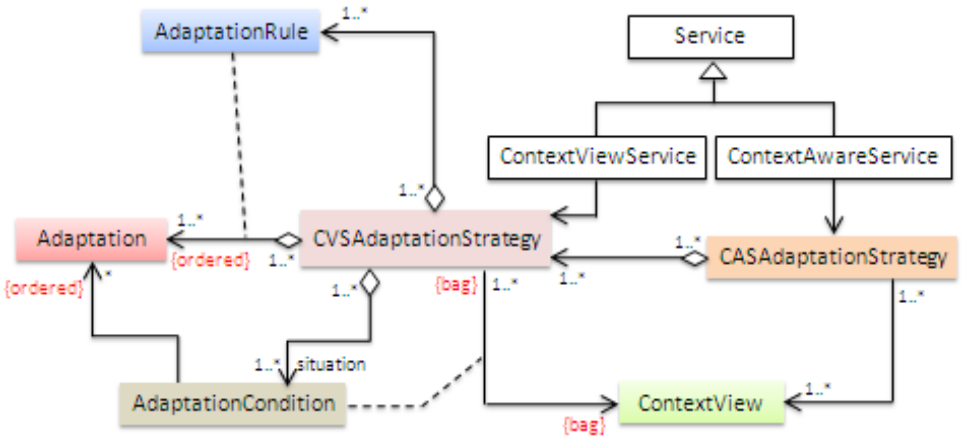

Fig. 6 Core CAS metamodel.

6. Aspect Based Pattern

7.1 Context-Awareness Mechanism

Traditional approaches used for CAS design and development present several problems. In fact, simple core service duplication for each ContextView is a software engineering anti-pattern (e.g., high-cost of maintenance), also integrating adaptation logics into core service makes it complex and decreases its ability to be reused and maintained. So, to rationalize the development and maintenance of CAS, we have to resort to new mechanisms and strategies that allow core service extension without any duplication or regression risks. Such mechanisms will favorite loosely coupling between the core service and its adaptations seen as crosscutting concerns.

Inspired by Separation of Concerns [23] and Aspect Paradigm concepts [6], our CAS design and development approach consists in considering Adaptation as an aspect. Thereby, the core service focuses only on business logic and all of its Adaptations related to its ContextViews will be defined separately as aspects called Adaptation Aspects. These Adaptation Aspects will be dynamically woven at runtime into the core service, by our tool named Adaptation Aspects Weaver (A^2W) (see Fig. 8), to produce the expected ContextViewService.

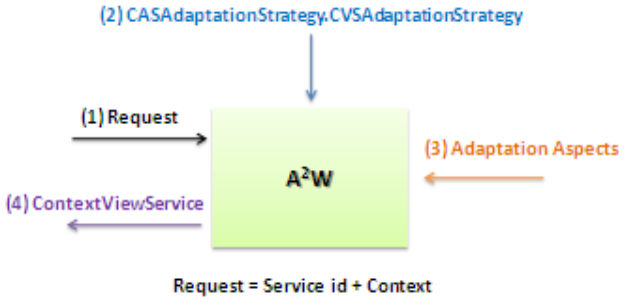

Fig. 8 Adaptation Aspects Weaver mechanism.

7.2 A^2W Architecture

Figure 9 illustrates the mechanism behind the A^2W tool. The *Request Notifier* notifies, in a synchronous or asynchronous mode, the *Decision Maker* with the executed service id and the execution context in purpose to recuperate the adequate CASAdaptationStrategy. Then, the *Decision Maker* inspects it in order to retrieve and interpret the current ContextView. The interpretation mechanism, operated by the Service Reconfigurator, consists in checking the AdaptationConditions to weave only the required Adaptation Aspects, following a set of AdaptationRules, into the core service to produce the corresponding ContextViewService.

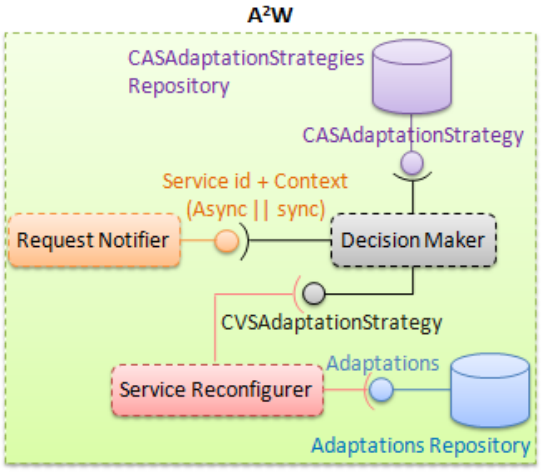

Fig. 9 A^2W architecture.

As shown in figure 10, once the tourist has requested a proposition of restaurants, the *Restaurants Controller* (i.e., the entry of the system in a MVC pattern) gets the context of the executing service from the *Context Manager*, and then forwards the *request* with the recuperated *context* to the *Request Notifier*. This last notifies the *Decision Maker* with the appropriate *serviceId*, *params* and *context*. Based on this information, the *Decision maker* retrieves the pertinent *CVSAdaptationStrategy* which will be used by the *Service Reconfigurer* in purpose to adapt the core service and provide a relevant response to the tourist expectations.

Figure 11 shows two views for the restaurant searching service depending on the context state. The full view, which contains full restaurants details, presents the nominal view, while the reduced view (only relevant information) presents the view returned by the service in the case of a low connection mode or battery level adaptation (restaurant photos are deleted from the response in order of optimization).

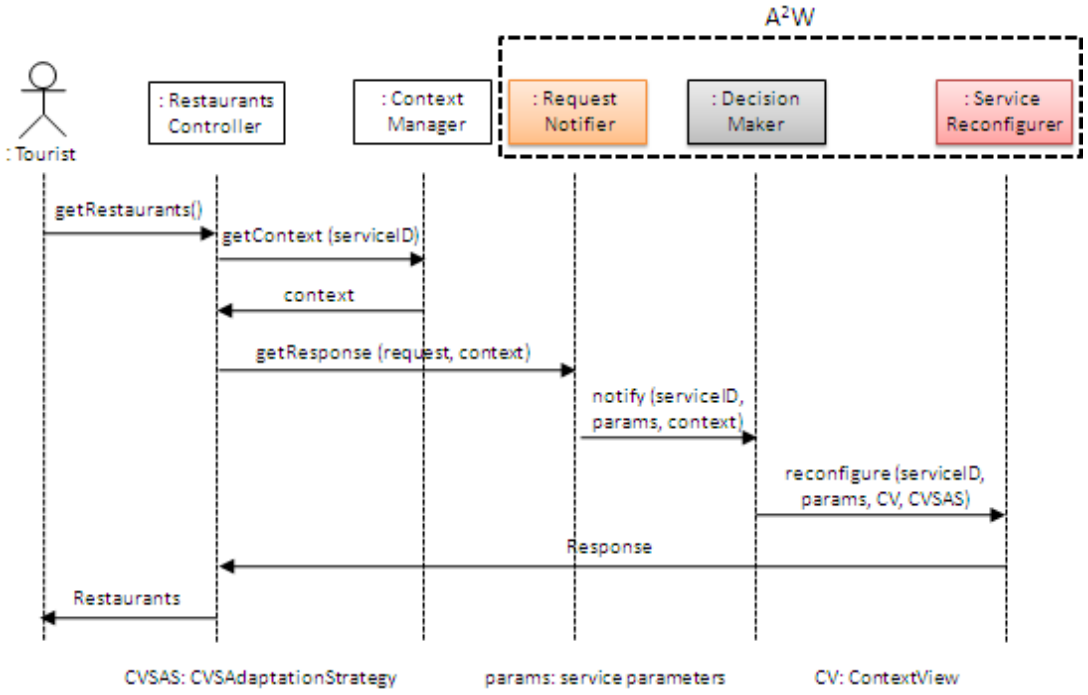

Fig. 10 Sequence diagram for the Restaurants Searching Service.

Fig. 11 Full and reduced views for the *Restaurants Searching* Service.

7.3 Tools and Frameworks Support

To develop our A^2W toll, we used the Eclipse and Xcode EDI with the following frameworks that respond to a specific technical and architectural purpose in our platform:

- Spring 2.5 [28] was used as IoC (Inversion of Control) container to link all the components of our framework, also, transaction is managed by this framework;
- Spring AOP 2.5 [28] represents the core framework of the A^2W tool in order to make the dynamic weaving for business service layer;
- Hibernate 3.3 [20] is the framework used in the persistence layer of the application to map the business model classes ;
- CXF 2.2 [14] is the soap middleware that manage all the communication purposes in our application using the web services technology;
- The final client of our system was developed using the iPhone SDK [15], the client application can be executed on iPhone and iPod devices;
- Configuration files written used XML technology is parsed using the JAXB2 OXM standard [21].

7. Related Work

Several context models have been defined (e.g., Key-value pairs [13], databases (e.g., CML [16]), ontologies (e.g., CMF [5]), profiling (e.g., CC/PP [8]), etc.) and various context-aware middleware and frameworks have been developed (e.g., context Toolkit [25], CoBrA [4], K-Components [26], CORTEX [13], etc.) to deal with context-aware applications development. In one side, the main objective of context modeling researches is to provide an abstraction of context information to permit easy context management and they do not deal, in general, with the adaptation of applications to the context. In the other side, researches that focus on frameworks and middleware development try to simplify context-aware applications development and they do not deal with the modeling of context-awareness of applications.

The rest of this section will focus on some other works that suggest the employment of model driven approaches for context-aware applications development. An important effort is the work conducted by Taconet and Kazi-Aoul in [7]. Authors define metamodels for modeling context-aware applications by planning several model views that model system context sensitivity, but they do not deal with adaptability. In our approach the service adaptability to the context is realized through the *CASAdaptationStrategy* artifact and the A^2W tool. Ayed [22] specify a MDD (Model Driven Development) approach and an UML profile to design context-aware applications independently of the platform. He also proposes a design process that models the contexts that impact an application and its variability but also does not deal with applications adaptation to the context. In ContextUML project [12], Sheng and Benatallah define an approach for modeling context-aware Web Services. Context in ContextUML is specialized into AtomicContext and CompositeContext, so the proposed metamodel does not refine context information. Moreover, authors do not specify the mechanism used to fulfill CAS adaptation. Another important domain concerns Product Line Engineering (PLE) that has a great potential in modeling service variability. An important work is the one conducted in CAPPUCINE project [9]. Authors focus on context-aware adaptation in Dynamic Service-Oriented Product Line (DSOPL) rather than context modeling, and propose two different processes for the initial and iterative phases of product derivation. The main challenge to be faced in this work is to reduce non-deterministic behaviors when non deterministic context-aware assets are introduced. In our work, this challenge is faced by the execution of an ordered set of adaptations.

8. Conclusion

In this paper, we proposed a context specification as a base for the context metamodel which is generic and open to allow its extension to various domains depending on needs. Then we presented a context provider metamodel which will serve for the context information acquisition. Finally, we proposed a CAS specification and metamodel and an adaptation mechanism that, based on the Aspect Paradigm, considers the adaptations of a service to its execution context as *Adaptation Aspects* dynamically woven by the A^2W tool.

We focused in this paper on proposing CAS artifacts metamodels for designing context-awareness of service oriented applications. In our future work, we project to include our metamodels (context, context provider, CAS) in the Eclipse Modeling Framework (EMF). Then use the Graphical Modeling Framework (GMF) to build a graphical editor that will allow designers to model CASs.

References

[1] H. Hafiddi, M. Nassar, H. Baidouri, B. El Asri and A. Kriouile, "*A Context-Aware Service Centric Approach for Service Oriented Architectures*", in the 13th International Conference on Enterprise Information Systems (ICEIS'11), Beijing, China, June 2011.

[2] H. Hafiddi, H. Baidouri, M. Nassar, B. El Asri and A. Kriouile, "*A Model Driven Approach for Context-Aware Services Development*",

in the 2nd International Conference on Multimedia Computing and Systems (ICMCS'11), Ouarzazate, Morocco, April 2011.
[3] H. L. Truong and S. Dustdar, "A survey on context-aware web service systems," *Int. J. of Web Information Systems*, vol. 5, no. 1, pp. 5-31, 2009.
[4] H. Chen, "An intelligent broker architecture for pervasive context-aware systems," Ph.D. thesis, Univ. of Maryland, Baltimore County, 2004.
[5] P. Korpipää and J. Mäntyjärvi, "An ontology for mobile device sensor-based context awareness," in the *4th Int. and Interdisciplinary Conf. on Modeling and Using Context*, vol. 2680 of LNCS, pp. 451-458, Springer, 2003.
[6] G. Kiczales et al., "Aspect-oriented programming," Proc. *European Conf. on Object-Oriented Programming*, vol. 1241 of LNCS, pp.220-242, Springer, June 1997.
[7] C. Taconet and Z. Kazi-Aoul, "Building context-awareness models for mobile applications," in *Digital Information Management J.*, vol. 8, no. 2, pp. 78-87, 2010.
[8] G. Klyne et al., "Composite Capability/Preference Profile (CC/PP): structure and vocabularies 2.0," W3C recommendation, Tech. Rep., Apr. 2007.
[9] C. Parra, X. Blanc and L. Duchien, "Context awareness for dynamic service-oriented product lines," in the *13th Int. Software Product Line Conf.*, San Francisco, USA, Aug. 2009.
[10] H. L. Truong and S. Dustdar, "Context coupling techniques for context-aware web service systems: an overview," in *Enabling Context-Aware Web Services: Methods, Architectures, and Technologies*, 1th ed.: Chapman and Hall/CRC, 2010, pp. 337-364.
[11] C. F. Sorensen et al., "Context-aware middleware for applications in mobile ad hoc environments," Proc. *2nd workshop on Middleware for pervasive and ad-hoc computing*, Toronto, Canada, Oct. 2004.
[12] Q. Z. Sheng and B. Benatallah, "ContextUML: A UML-based modeling language for model-driven development of context-aware web services," Proc. *4th Int. Conf. on Mobile Business*, Sydney, Australia, July 2005, pp. 206-212.
[13] B. N. Schilit, M. M. Theimer and B. B. Welch, "Customizing mobile applications," Proc. *USENIX Symp. Mobile and Location-Independent Computing*, Cambridge, MA, Aug.1993, pp. 129-138.
[14] http://cxf.apache.org/.
[15] http://developer.apple.com/devcenter/ios/index.action.
[16] K. Henricksen and J. Indulska, "Developing context-aware pervasive computing applications: models and approach," in *Pervasive and Mobile Computing J.*, Elsevier, vol. 2, no. 1, pp. 37-64, Feb. 2006.
[17] B. Schilit and M. Theimer, "Disseminating active map information to mobile hosts," *IEEE Network*, vol. 8, no. 5, pp. 22-32, Sep./Oct. 1994.
[18] Enabling Context-Aware Web Services: Methods, Architectures, and Technologies. Chapman and Hall/CRC, Ed. 2010, pp. 113-135.
[19] P. Brezillon, "Focusing on context in human-centered computing," *IEEE Intelligent Syst.*, vol. 18, no. 3, pp. 62-66, May 2003.
[20] http://www.hibernate.org/.
[21] http://jaxb.java.net/.
[22] D. Ayed, D. Delanote and Y. Berbers, "MDD approach for the development of context-aware applications," in the *6th Int. and Interdisciplinary Conf. on Modeling and Using Context*, vol. 4635 of LNCS, pp.15-28, Springer, 2007.
[23] W. Hürsch and C. V. Lopes, "Separation of concerns," Northeastern Univ., Boston, Mass., Tech. Rep. NUCCS-95-03, Feb. 1995.
[24] M. P. Papazoglou, "Service oriented computing: concepts, characteristics and directions," in *Information Syst. J.*, *IEEE Comput. Soc.*, vol. 50, no. 2, pp. 3-12, Dec. 2003.
[25] D. Salber, A. K. Dey and G. D. Abowd, "The Context Toolkit: aiding the development of context-enabled applications," Proc. *SIGCHI Conf. on Human Factors in Computing Syst.*, Pittsburgh, PA, USA, May 1999, pp. 434-441.
[26] J. Dowling and V. Cahill, "The K-Component architecture meta-model for self-adaptive software," Proc. *Reflection'01*, vol. 2192 of LNCS, pp. 81-88, Springer, Sep. 2001.
[27] A. Schmidt, M. Beigl and H. W. Gellersen, "There is more to context than location," in *Computers and Graphics J.*, Elsevier, vol. 23, no. 6, pp. 893-902, Dec. 1999.
[28] http://www.springsource.org/.
[29] A. K. Dey and G. D. Abowd, "Towards a better understanding of context and context-awareness," GVU Center, Georgia Inst. of Technology, Tech. Rep. GIT-GVU-99-22, June 1999.

Hatim Hafiddi received the Engineer of state degree in Software Engineering from National High School of Computer Science and Systems Analysis (ENSIAS) in 2007. He is currently a PhD student in the IMS (Models and Systems Engineering) Team of SIME Laboratory at ENSIAS. His research interests are Context-Aware Service-Oriented Computing, Aspect Oriented Engineering, Mobile Information Systems Engineering, and Model-Driven Engineering.

Hicham Baidouri received the Engineer of state degree in Software Engineering from Mohammadia School of Engineers (EMI) in 2007. He is currently a PhD student in the IMS (Models and Systems Engineering) Team of SIME Laboratory at ENSIAS. His research interests are Context-Aware Service-Oriented Computing, Aspect Oriented Engineering, Mobile Information Systems Engineering, and Model-Driven Engineering.

Mahmoud Nassar is Professor and Head of the Software Engineering Department at National Higher School for Computer Science and Systems Analysis (ENSIAS), Rabat, Morocco. He is also Head of IMS (Models and Systems Engineering) Team of SIME Laboratory. He received his PhD in Computer Science from the INPT Institute of Toulouse, France. His research interests are Context-Aware Service-Oriented Computing, Component based Engineering, and Model-Driven Engineering. He leads numerous R&D projects related to the application of these domains in Embedded Systems, e-Health, and e-Tourism.

Abdelaziz Kriouile is a full Professor in the Software engineering Department and a member of SI2M Laboratory at National Higher School for Computer Science and Systems Analysis (ENSIAS), Rabat. He is also a Head of the SI3M Formation and Research Unit. His research interests include integration of viewpoints in Object-Oriented Analysis/Design, Service-Oriented Computing, and speech recognition by Markov models. He has directed several Ph.D thesis in the context of Franco-Moroccan collaborations.