Towards a Context-Aware Composition of Services*

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

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

Summary
Over the last few years, recent evolutions in wireless technologies and intelligent mobile devices have enabled the emergence of mobile services. With such services, context plays great role in their adaptation in order to produce the most appropriate behavior. Using the SOA paradigm, some service may be achieved through the cooperation of several services to perform more complicated functionality. These composite services must be designed in a manner that allows them to handle certain level of context awareness in order to provide the most suitable end-user experience. Our focus in this paper is to present our proposal of adapting service composition by the integration of context during the composition process. This dynamic context-aware composition of services is realized through our Context-Aware Composition Builder tool.

Key words:
Context-aware composite service; model driven engineering; context-aware compostion.

1. Introduction

Nowadays, one of the most desired features of end-users from services providers is the possibility to get adapted composite services to their preferences and needs. This process, known as contextual adaptation, requires from the composite services to exploit past and present information about the user such as his current situation, his location, his profile, etc. This commitment has involved the introduction of a new type of composite services named Context-Aware Composite Services (CACS). In order to be context-aware, composite services need to follow some requirements in order to resolve the challenges brought by the context-awareness paradigm. First, the composition technique of the existing service should be platform-agnostic, so the used approach could be projected on any technologies and implementation tools. Second, the composed service should be built in dynamic way depending on the context of use, i.e., the expected service must be generated at the execution stage. Compared to traditional service composition approaches, context aware composition computing emphasizes more open-endedness in terms of analysis, design and implementation phases.

CACS development can profit from existing paradigms and technologies such as process orchestration language (e.g., BPEL [6]) and Model Driven Engineering (MDE [22]). Process orchestration languages are very developed tool that enable the transparency and ease of use of the creation of composed service in dynamic way aiming at extending application functionalities. In our approach, BPEL descriptions of CACS are dynamically generated, through our Context-Aware Composition Builder tool, to provide the most suitable service composition regarding the current user context. MDE is a model centric approach for software development, in which models are used to drive software development life cycle. In our approach, CACS artifacts meta-models are provided to guide the design of CACS models, then, the implementation is generated automatically by performing a series of model to model transformations.
The rest of this paper is organized as follows. We present in next section a scenario that concerns an E-tourism system and highlight the context-awareness challenges. In Sect. 3, we present our context metamodel. Sect. 4 presents our CACS metamodel. We present, in Sect. 5, our Context-Aware Composition Builder tool. Sect. 6 briefly compares related work. Finally, we conclude the paper in Sect. 7 with plans for future work.

2. E-tourism Motivating Scenario: Tourism Tour Service

The following motivating scenario relates to a context-aware e-tourism system. It aims to help the out-of-towners who need some guidance (i.e. tour planning) on how they will spend their free time in a foreign city (see Fig. 1).

Let’s say that a tourist wants to discover the history, culture, monuments, landscapes and gastronomy of a foreign city. So, he accesses a context-aware e-tourism system, offered by a local provider, using his mobile device (e.g., PDA, Smartphone, Tablet, etc.). This system will suggest a complete tour of the city for an entire day or just for a specific period of the day (e.g., morning, evening, etc.) depending on the tourist free time. Furthermore, the tour sent back to the tourist will take into account other context information such as time and weather parameters (e.g., in summer, the system will favor beaches over monuments), user profile (e.g., probably append a party at

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

Manuscript received March 5, 2012
Manuscript revised March 20, 2012

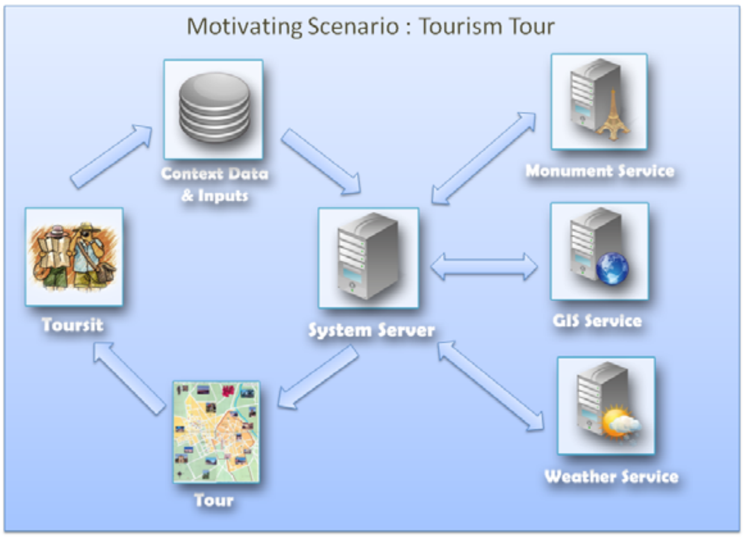


Fig. 1 Involved services in the "Tourism Tour scenario".

the end of the tour if the user has mentioned it in his preferences) and the used mobile device (e.g., configuration, CPU, resolution, etc.) in order to improve the user experience and sent the most suited response. Likewise, the system will propose the transport (i.e., GIS service) between each places of the proposed tour and display all the possible alternatives (e.g., subway, bus, taxi, etc.) depending on the distance, the weather and the tourist needs.
This e-tourism scenario highlights the fundamental challenges for the development of context-aware composition of services in context-aware systems. First, context definition (i.e., which context information are relevant for an adequate composition of services) and acquisition is not an evident process. Second, the composition process must be realized in a dynamic way depending on the execution context. By way of illustration, the previous scenario highlights the two following dynamic compositions, of the tour planning service, depending of the user context:

- Suppose that the tourist is visiting the city in summer, the system should compose the tour starting with beach in the morning (using a partner e-tourism service), then propose the suited restaurant for lunch, program a monument visit at the evening, and according to user preference, append a party animation at the night to the program.

- Assume that another tourist is visiting the city in spring, the system should propose natural landscapes instead of beach, and the rest of the tour could change depending on the user needs, weather and city transport infrastructure.

3. Context

Context is the information that characterizes the interactions between humans, applications, and the environment [11]. Several context definitions were proposed in the literature (e.g., [19], [20], [10], [21], [2], 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*" [23].

In our approach we choose to use the context metamodel developed in [5] for different reasons. Rather than giving a domain specific formalization of context this metamodel is domain and platform independent, and can be extended, if needed, to support various domains. This core context metamodel (see Fig. 2) specify a context as 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).

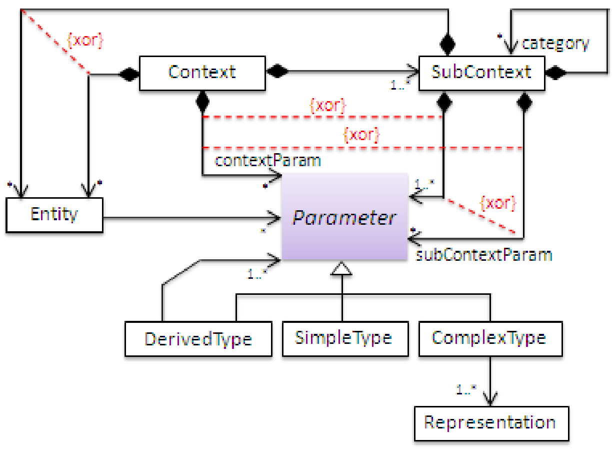


Fig. 2 Core context metamodel [5].

4. Context-Aware Composite Service

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 [17]. The vision of service as a software component allows combining several services, providing a global value-added service, called composite service. A context-aware composition of services (i.e., context-aware composite service) is a composition which is able to present different configurations according to the execution context named context view [3] (see Fig. 3). In our approach, a context-view composite service presents the result of an adapted composite service to a given context view, and the various context-view composite services for a given composite service forms the context-aware composite service.

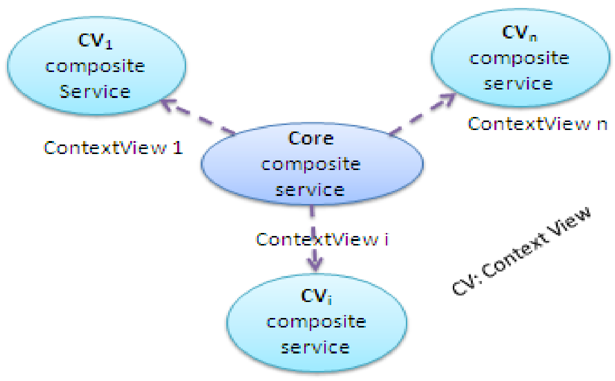


Fig. 3 Core composite service adaptation to its various contextviews.

Fig. 4 illustrates our Context-Aware Composite Service metamodel. This metamodel is based on the following specification:

- All of Elementary Service, Composite Service, Context-View Composite Service (i.e., *CVCompositeService*) and Context-Aware Composite Service (i.e., *CAComposite Service*) are specific services;
- A context-aware composite service possesses a context-aware composition strategy (i.e., *CACompositionStrategy*) which concerns a set of context views;
- A context-view composite service possesses a context-view composition strategy (i.e., *CVCompositionStrategy*) which concerns a given context view;
- A context-aware composition strategy aggregates a set of context-view composition strategies;
- For a given context-view composition strategy and context view, a set of configuration conditions (i.e., *ConfigCondition*) is deduced;
- A configuration condition may involve a set of services configuration;
- For a given context-view composition strategy and service, a configuration rule (i.e., *ConfigRule*) is associated;
- A context-view composition strategy aggregates a set of configuration conditions, configuration rules and services.

In our specification, a context-aware composite service is seen as a specific composite service with a number of *ContextViews*. For each one, we associate a context-view composition strategy (i.e., *CVCompostionStrategy*) which indicates when (i.e., *ConfigCondition*: classical condition expressed on *ContextView* parameters) and how (i.e., *ConfigRule*: defines how the configuration (i.e., the execution chronology and the types of dependencies) must be realized in the core composition) a set of services (i.e., Service) cooperates in order to provide the expected composition regarding the current execution context.

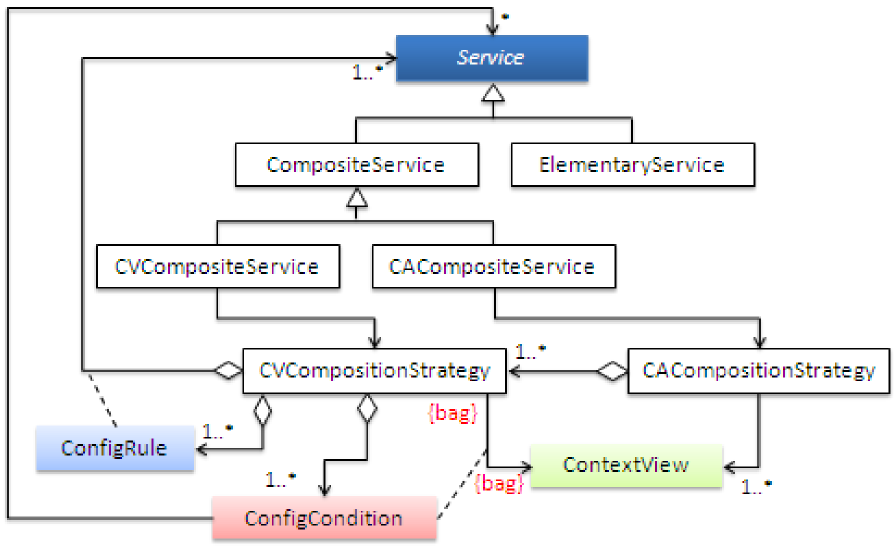
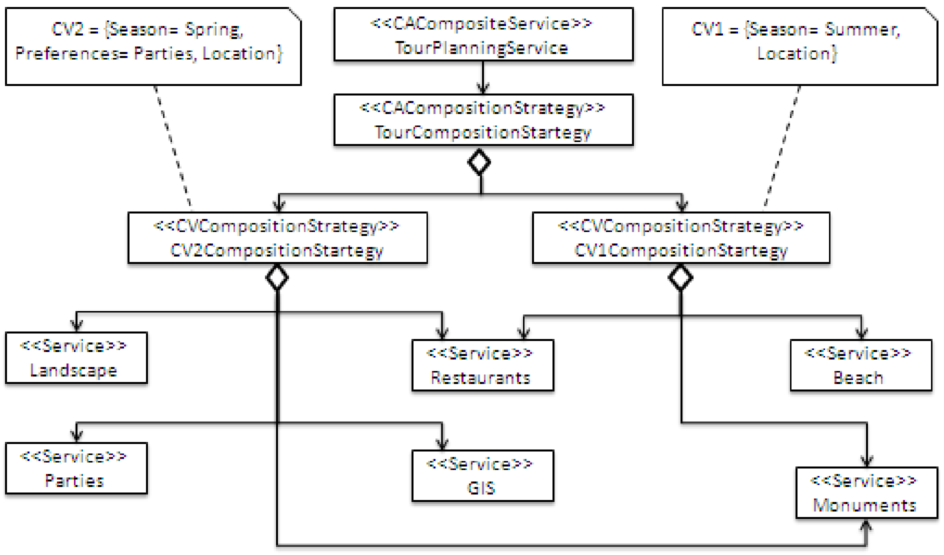


Fig. 4 Core Context-Aware Composite Service metamodel.

The composition result forms the context view composite service (i.e., *CVCompositeService*). So, for a given composite service, the set of its *CVCompositeServices* (respectively *CVCompositionStrategies*) forms the *CACompositeService* (respectively *CACompositionStrategy*).

As illustrated in Fig. 5, involved services in the tour planning composite service may change depending to the context parameters and their values.



CV2 = {Season= Spring, Preferences= Parties, Location}
<<CACompositeService>> TourPlanningService
CV1 = {Season= Summer, Location}
<<CACompositionStrategy>> TourCompositionStartegy
<<CVCompositionStrategy>> CV2CompositionStartegy
<<CVCompositionStrategy>> CV1CompositionStartegy
<<Service>> Landscape
<<Service>> Restaurants
<<Service>> Beach
<<Service>> Parties
<<Service>> GIS
<<Service>> Monuments

Fig. 5 Succinct CACompositeService model for the Tourism Tour Service.

5. Context-Aware Composite Service

5.1 Architecture

Today, it is very clear that classical approaches for context-aware composition development present several limitations. Indeed, designing composite service variant for each context-view or introducing all composition scenarios in the same composite is, deeply, a software engineering anti-pattern (e.g., high-cost of maintenance).
So, to rationalize the development and maintenance of context-aware composite services, we have to resort to a strategy pattern that allows dynamic composition without any duplication or regression risks. Our strategy reposes on a NDC (Notify, Decide, Configure) pattern which is implemented by our Context-Aware Composition Builder tool.
Fig. 6 illustrates the mechanism behind the Context-Aware Composition Builder tool. So, The Request Notifier notifies, in a synchronous or asynchronous mode, the Decision Maker with the executed composite service id and the execution context in purpose to recuperate the adequate CSCompositionStrategy. Then, the Decision Maker inspects it in order to retrieve and interpret the current ContextView.

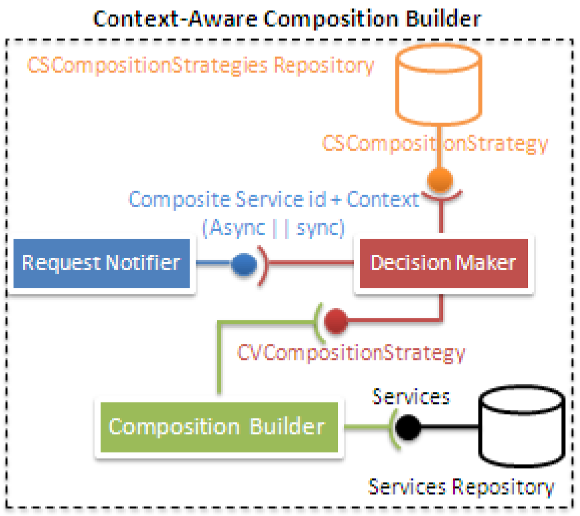


Fig. 6 Context-Aware Composition Builder Tool.

The configuration mechanism, operated by the Composition Builder, consists in checking the ConfigConditions to build dynamic service composition, following a set of ConfigRules, to produce the corresponding CVCompositeService.

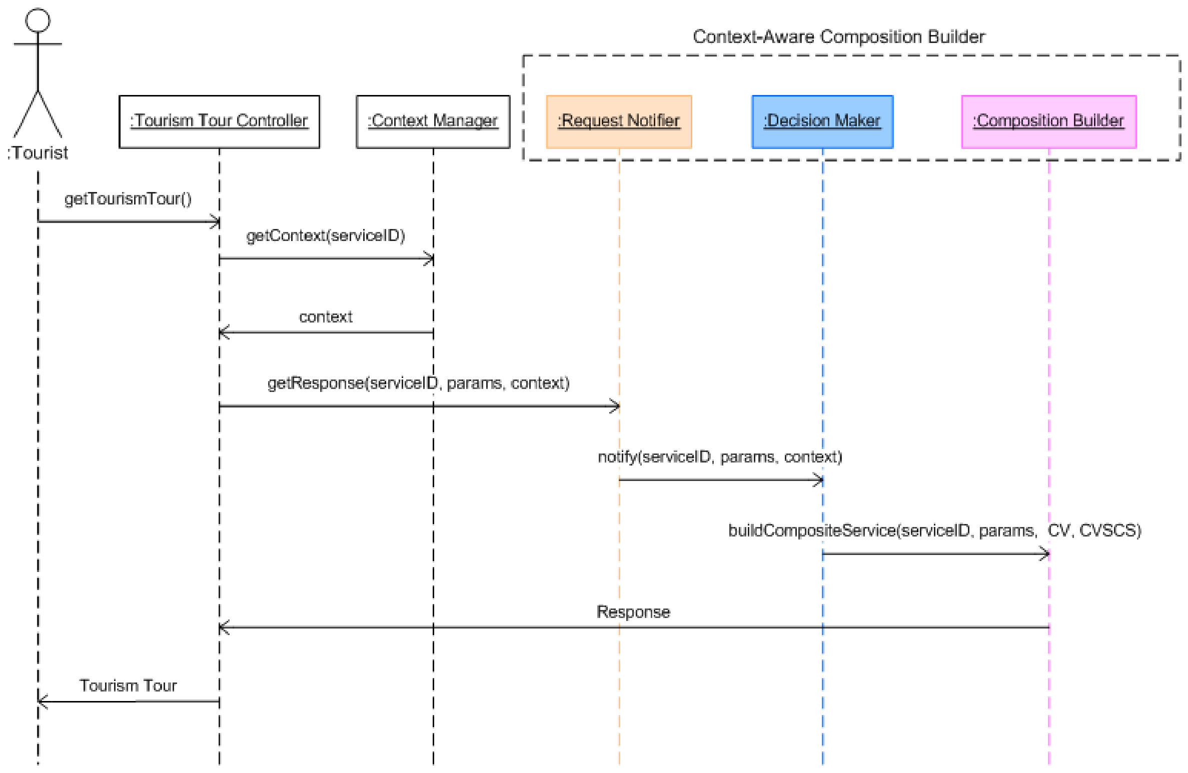


Fig. 7 Sequence diagram for the Tourism Tour Service.

As shown in figure 7, once the tourist asks the system for the tourism tour, the Tourism Tour Controller (i.e. the entry point of the system using MVC pattern) retrieves the context of the requested service using the Context Manager, then forwards the request composed from the serviceID, service params and the resulted context to *Request Notifier* (this represents the façade of our tool). The next step consists of notifying the *Decision Maker* with the appropriate serviceID, params and context. Based on this, the *Decision Maker* recovers the suited *CVSCompositionStrategy* that will be used by the *Composition Builder* in order to generate the execution chronology of the composite service and send the expected tourism tour to the user.

5.2 Tools and Frameworks Support

To develop our Context-Aware Composition Builder tool, we used the Eclipse EDI with the following frameworks that respond to a specific technical and architectural purpose in our platform:

- Spring 2.5 [18] was used as IoC (Inversion of Control) container to link all the components of our framework, also, transaction is managed by this framework.
- Hibernate 3.3 [12] is the framework used in the persistence layer of the application to map the business model classes.
- CXF 2.2 [8] is the soap middleware that manage all the communication purposes in our application using the web services technology.
- Configuration files written used XML technology is parsed using the JAXB2 OXM standard [13].
- We used Apache ODE [15] as the BPEL engine in order to generate the expected composition service. This tool allows the execution of one or more business web services expressed using the Web Services Business Process Execution Language (WS-BPEL). It principally communications with services by sending and receiving messages, manipulating data and handling exceptions as defined by any given process. Also, the engine supports the HTTP WSDL for binding, allowing invocation of REST-style web services.

6. Related Work

In this section, we will deal with a representative subset of existing studies that work on context-aware composition mechanisms to emphasize the similarities and differences with our approach.

Context aware service composition process is entangled with several complex features such as context modeling, context retrieving, service adaptation and orchestration. The composition mechanism can happen in different time of the development process, some existing works consider the composition logic at the deployment time like the context aware tool CADeComp [4]. The metamodel used in this project is based on OMG D&C specifications [9], and follows MDA specifications. The CADeComp project describes context aware assemblies of components and produces target deployment plan. At the deployment time, a set of adaptation rules is executed based on the corresponding context adaptation. Likewise, PLASTIC project [1] presents similar concept to CADeComp providing several tools and methodologies to develop service-based context aware applications. In this work, authors introduced a new metamodel based on two levels of software description: service composition as an abstract layer and component compositions as a concrete layer where deployed services exist. Context information is mainly utilized at the service discovery step in order to perform the expected composite service. In ContextUML [7], authors introduced a dedicated UML metamodel which extends the existing UML syntax by introducing appropriate artifacts in order to enable the creation of context-aware service and composite models. In this work, context entity dives into two categories (state-based and event-based). Each context type is associated with a constraint that performs an action during the composition time. Context information can be either used to be mapped to specific values or modify the structure or the application behavior. The main challenge to be faced in this work is to reduce non-deterministic behaviors when non-deterministic context-aware assets are introduced.

Other context aware composition studies use the middleware programming paradigm; the expected composite service is twisted from unitary service and/or composite service. The MySIM [14] is one of this middlewares that integrate services in a transparent way using the OSGi/Felix platform. It uses the reflexive techniques to do the syntactic interface matching and ontology online reasoner for the semantic matching. The technique is interesting but solutions need to be found to make the spontaneous service integration scalable to large environments. Another platform similar to MySIM is the PERSE project [16]. Four modules present the main essence of this project, however the Evaluator module responsible of computing the suited composition combination is the most developed component of the project. The efficiency of PERSE has been tested and proved in the cost evaluation in terms of service matching, service composition and processing time for service composition. Most of the middleware context-aware composition approach presents only two granularities of services unitary (or classic) service and component service to generate the expected behavior. The re-use of the context aware composite service at the composition level is not taken in charge. In concerns with the above mentioned approaches all the development stages (analysis, design and implementation) of the system take care of context in dynamic way. Additionally, context modeling, retrieving and handling phases are independents from the base application functionality. Focus is given only to the service functional design and application flow that indicates the order in which services are invoked regarding the context state.

7. Conclusion and Perspectives

In this paper, we have proposed a context-aware composition specification as a base for the context-aware composite service metamodel. Then, we presented our Context-Aware Composition Builder tool which reposes in a NDC pattern and exploit our context-aware composite service metamodel in order to provide, in a well design way, dynamic composition of services depending to the current user context.

We focused in this paper on proposing context-aware composite service metamodel for designing context-aware composite services and a Context-Aware Composition Builder tool. In our future work, we project to include our metamodel in the Eclipse Modeling Framework (EMF). Then use the Graphical Modeling Framework (GMF) to build a graphical editor that will allow designers to model context-aware composite services. Finally, we will implement transformations using Query/View/Transformation (QVT) in order to transform from CACS technology independent models to the specific models, and use MOF script for generating executable code.

References

[1] M. Autili, V. Cortellessa, A. D. Marco and P. Inverardi, “A conceptual model for adaptable context-aware services” Proc. of Int. Workshop on Web Services Modeling and Testing (WS-MaTe2006), pp. 15-33, Palermo, Sicily, ITALY, June 9th 2006.

[2] A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, K. V. Laerhoven and W. V. Velde, “Advanced Interaction in Context”, In HUC ’99 : Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, pp. 89-101, London, UK, 1999. Springer-Verlag.

[3] 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.

[4] D. Ayed, C. Taconet, G. Bernard and Y.Berbers, “An adaptation methodology for the deployment of mobile component-based applications”, In IEEE Int. Conf. on Pervasive Services (ICPS’06), pp. 193-202, Lyon, France, June 2006.

[5] H. Hafiddi, H. Baidouri, M. Nassar and A. Kriouile, “An Aspect Based Pattern for Context-Awareness of Services”, Int. J. of Comp. Science and Network Security, vol. 11, no. 12, 2011.

[6] OASIS. Business Process Execution Language (BPEL) 2.0. 2007. Available at: http://docs.oasisopen.org/wsbpel/2.0/wsbpel-v2.0.html.

[7] 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.

[8] http://cxf.apache.org/.

[9] Object Management Group, “Deployment and Configuration of Component-based Distributed Applications”, June 2003. Draft Adopted Specification (ptc/03-07-02).

[10] B. Schilit and M. Theimer, “Disseminating active map information to mobile hosts”, IEEE Network, vol. 8, no. 5, pp. 22-32, Sep./Oct. 1994.

[11] P. Brezillon, “Focusing on context in human-centered computing,” IEEE Intelligent Syst., vol. 18, no. 3, pp. 62-66, May 2003.

[12] http://hibernate.org/.

[13] http://jaxb.java.net/.

[14] N. Ibrahim, F. Le Mouël and S. Frénot, “MySIM: a Spontaneous Service Integration Middleware for Pervasive Environments”, In ACM International Conference on Pervasive Services (ICPS'2009), 2009, London, UK.

[15] http://ode.apache.org/.

[16] S. Ben Mokhtar, “Semantic Middleware for Service-Oriented Pervasive Computing”, Ph.D. thesis, University of Paris 6, Paris, France, 2007.

[17] 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.

[18] http://springsource.org/.

[19] 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.

[20] 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.

[21] P. J. Brown, “The stick-e document: a framework for creating context-aware applications”, Proc. of the Electronic Publishing, Palo Alto, pp. 259-272, 1996.

[22] J. M. Favre, “Towards a Basic Theory to Model Driven Engineering”, UML 2004 – Workshop in Software Model Engineering (WISME 2004), 2004.

[23] 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.

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.

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.

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.