A Mediator Architecture for Context-aware Composition in SOA

Hicham Baidouri, Hatim Hafiddi, Mahmoud Nassar and Abdelaziz Kriouile
IMS Team, SIME Laboratory, ENSIAS, Rabat, Morocco

Keywords: Context, Ubiquitous Computing, Context-aware Service, Context-aware Composite Service, Aspect Paradigm, Context-Aware Composition, Model Driven Engineering.

Abstract: The emergence of wireless technologies, intelligent mobile devices and service oriented architectures has enabled the development of the context-aware service oriented systems. This evolution has put the light on a challenging problem: how to dynamically compose services in SOA based systems to perform more complicated functionalities and provide richer user experience? As observed from the literature, several researches focus mainly on context-aware service design and modelling, but few studies have worked on the composition of this new kind of service to provide more complicated features. In this paper, we aim 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 Mediator Architecture for Context-Aware Composition (MACAC).

1 INTRODUCTION

Over the last few years, adaptive service composition has emerged as one of the most desired features that allow the integration and cooperation between pre-existing services to bring out new features. This process, known as contextual service composition, requires from the composite services to consider information from the user's context – such as location, profile, age, etc – by performing several adaptations on service behaviour in first stage and composition technique in second stage. The main challenge is to operate the most suitable service combination in order to respond to user expectation and improve the end-user experience. This obligation 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 (OASIS, 2007)) and Model Driven Engineering (MDE (Favre, 2004)). Process orchestration languages are very developed tool that enable the transparency and ease of use of the creation of composed service aiming at extending application functionalities. In our approach, BPEL descriptions of CACS are dynamically generated, through our MACAC 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 artefacts meta-models are provided to guide the design of CACS models, then, the implementation can be 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 and context provider metamodels for context management. Sect. 4 presents our CACS specification and metamodel. We present, in Sect. 5,

Baidouri H., Hafiddi H., Nassar M. and Kriouile A..
A Mediator Architecture for Context-aware Composition in SOA.
DOI: 10.5220/0003992502450251
In *Proceedings of the 14th International Conference on Enterprise Information Systems* (ICEIS-2012), pages 245-251
ISBN: 978-989-8565-11-2

our MACAC mediator for context-aware composition of services. Sect. 6 briefly compares related work. Finally, we conclude the paper in Sect. 7 with plans for future work.

2 e-TOURISM SCENARIO

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).

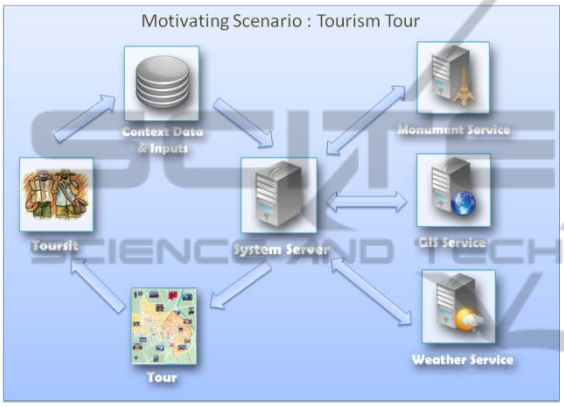

Figure 1: Involved services in the "Tourism Tour scenario".

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 favour beaches over monuments), user profile (e.g., probably append a party at 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 on 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 MANAGEMENT

3.1 Context

Context is the information that characterizes the interactions between humans, applications, and the environment (Brezillon, 2003). Several context definitions were proposed in the literature (e.g., (Salber, Dey and Abowd, 1999), (Schmidt, Beigl and Gellersen, 1999), (Schilit and Theimer, 1994), (Brown, 1996), (Schmidt et al., 1999), 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*" (Dey and Abowd, 1999).

In our approach we choose to use the context metamodel developed in (Hafiddi et al., 2011) 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).

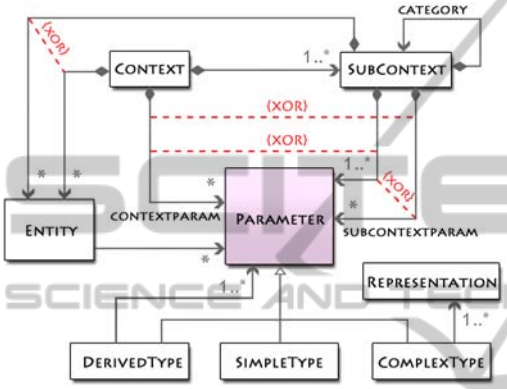

Figure 2: Core context metamodel.

3.2 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 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.

In our specification (Hafiddi et al., 2011), 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

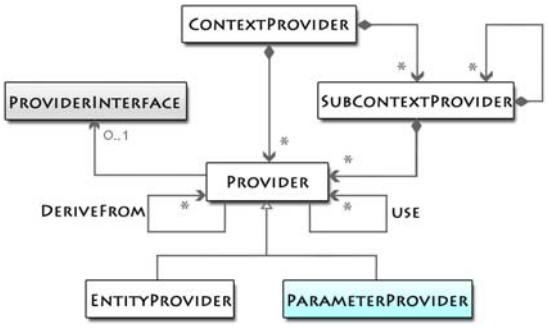

Figure 3: Core context provider metamodel.

parameters providers may 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 or derive from a set of providers. For example, a weather provider uses the localization provider to get the weather information.

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 (Papazoglou, 2003). 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 *ContextView* (Hafiddi et al., 2011) (see Fig.4)

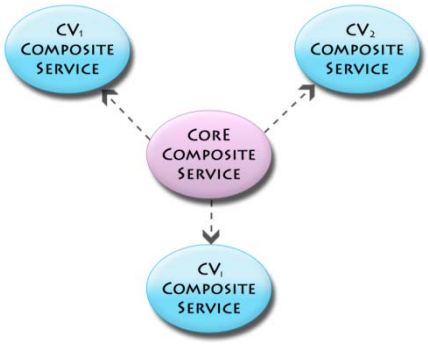

Figure 4: Core composite service adaptation to its various ContextViews.

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.

Figure 5 illustrates our Context-Aware Composite Service metamodel. This metamodel is based on the following specification:

- Elementary Service, Context-Aware 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 service is able to adapt dynamically its behaviour to its several execution (i.e., use) contexts. In other words, a context-aware service possesses mechanisms in purpose to exploit only relevant information of the execution context and adapt dynamically its behaviour (refer to (Hafiddi et al., 2011) for more details about context-aware services specification);
- 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. 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 figure 6, involved services in the tour planning composite service may change depending to the context parameters and their values.

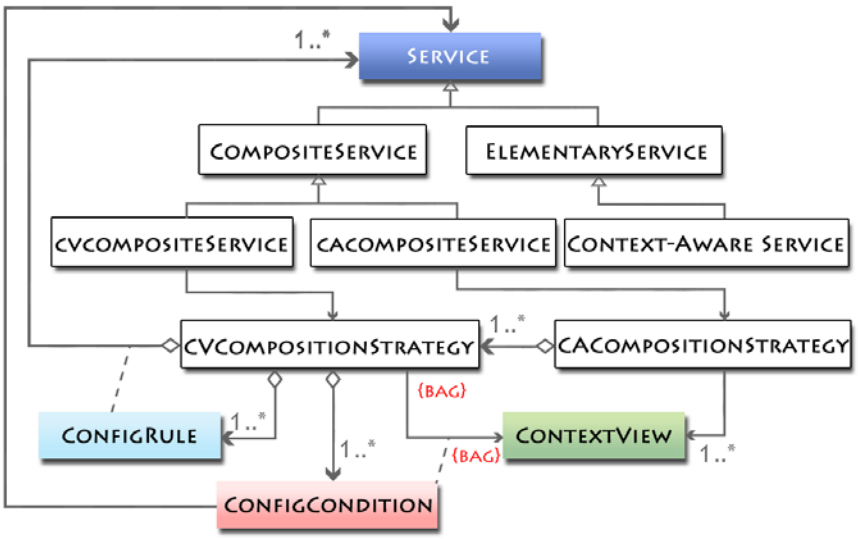

Figure 5: Core CACS metamodel.

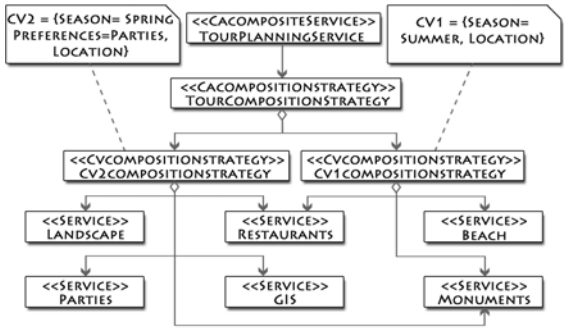

Figure 6: Succinct CACS model for the Tourism Tour Service.

5 CONTEXT-AWARE COMPOSITION MECHANISM

5.1 Mediator 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 and Configure) pattern which is implemented by the MACAC mediator.

Figure 7 presents the core components of our MACAC tool. As an entry point, the *RequestNotifier* takes care of each request coming to the platform. The next step is to check whether the type of the requested service is elementary or composite using the *ServiceExaminator* entity:

- In the case of an elementary service, the generation process of the context aware service is similar to the generation mechanism performed by the A2W tool already described in (Hafiddi et al., 2011). The *ESHandler* plays the same role as the *RequestNotifier* of the A2W tool.
- In the case of a composite service, the *CSHandler* receives the request (service id and context data) that will be passed to the *CompositionDecisionMaker* responsible of *CSCompositionStrategy* recuperation from the current *ContextView*. After that, the *CompositionBuilder* analyzes each *CSCompositionStrategy* and evaluate eventually the corresponding conditions in order to extract the best composition combination of services. Note that, the *CompositionBuilder* could invoke the *ESHandler* or/and *CSHandler* to build the necessary elementary or/and composite context-aware service for the global service generation.

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 (SpringSource, 2007) was used as IoC (Inversion of Control) container to link all the components of our framework, also, transaction is managed by this framework;

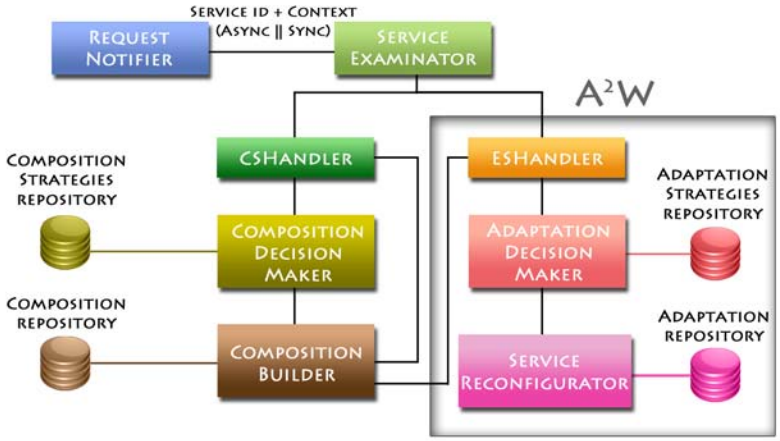

Figure 7: MACAC architecture.

- Hibernate 3.3 (Red Hat, 2011) is the framework used in the persistence layer of the application to map the business model classes;
- CXF 2.2 (Apache CXF, 2011) 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 (Oracle, 2006);
- We used Apache ODE (Apache ODE, 2011) 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 modelling, 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 (Ayed et al., 2006). The metamodel used in this project is based on OMG D&C specifications (OMG, 2003), 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 (Autili et al., 2006) 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.

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 (Ibrahim, Le Mouël and FreÏnot, 2009) is one of these 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 (Ben Mokhtar, 2007). 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 behaviour. 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 modelling, 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 CONCLUSIONS

In this paper, we aimed to propose a specification for context-aware composition system. So, we started by presenting our motivating scenario relative to an e-tourism context-aware system, which consist of providing a tour guide based on existing services. Then we presented a generic approach for modelling and collecting context information, which presents the basis for the elementary and composite context-aware service in terms of design and development.

Furthermore, we focused on proposing a meta-model for designing context-aware composite services, this meta-model was defined to enable the reuse of all the following type of services: elementary service, elementary composite service, context-aware service and context-aware composite service. Finally, we advanced our MACAC tool based on BPEL technology responsible of the dynamic generation of the expected composite service. In our future work, we project to include our meta-model in the Eclipse Modelling Framework (EMF). Then use the Graphical Modelling 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

Apache CXF (2011). *http://cxf.apache.org/*.

Apache ODE (2011). *http://ode.apache.org/*.

Autili, M., Cortellessa, V., Marco, A., D. and Inverardi, P. (2006). *A conceptual model for adaptable context-aware services*. In Proceedings of international Workshop on Web Services Modeling and Testing (WS-MaTe2006), pp. 15-33, Palermo, Sicily, Italy.

Ayed, D., Taconet, C., Bernard, G. and Berbers, Y. (2006). *An adaptation methodology for the deployment of mobile component-based applications*. In IEEE International Conference on Pervasive Services (ICPS'06), pp. 193-202, Lyon, France.

Ben Mokhtar, S. (2007). *Semantic Middleware for Service-Oriented Pervasive Computing*. In Ph.D. thesis, University of Paris 6, Paris, France.

Brezillon, P. (2003). *Focusing on context in human-centered computing*. IEEE Intelligent Systems, 18(3), 62-66.

Brown, P., J. (1996). *The stick-e document: a framework for creating context-aware applications*. In Proceedings of the Electronic Publishing, Palo Alto, pp. 259-272.

Dey, A., K. and Abowd, G., D. (1999). *Towards a Better Understanding of Context and Context-Awareness.* In Technical Report GIT-GVU-99-22, GVU Center, Georgia Institute of Technology.

Favre, J., M. (2004). *Towards a Basic Theory to Model Driven Engineering*. In Workshop in Software Model Engineering (WISME 2004).

Hafiddi, H., Nassar, M., Baidouri, H., El Asri, B. and Kriouile, A. (2011). *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.

Hafiddi, H., Baidouri, H., Nassar, M., El Asri, B. and Kriouile, A. (2011). *A Model Driven Approach for Context-Aware Services Development*. In the 2nd International Conference on Multimedia Computing and Systems (ICMCS'11), Ouarzazate, Morocco. IEEE Computer Society.

Ibrahim, N., Le Mouël, F. and FreÏnot, S. (2009). *ìMySIM: a Spontaneous Service Integration Middleware for Pervasive Environments*. In ACM International Conference on Pervasive Services (ICPS'2009), London, UK.

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

Object Management Group (2003). *Deployment and Configuration of Component-based Distributed Applications*. Draft Adopted Specification (ptc/03-07-02).

Oracle (2006). *http://jaxb.java.net/*.

Papazoglou, M., P. (2003). *Service Oriented Computing: Concepts, Characteristics and Directions*. In WISE'03, the 4th International Conference on Web Information Systems Engineering. IEEE Computer Society, 3-12.

Red Hat (2011). *http://hibernate.org/*.

Salber, D., Dey, A., K. and Abowd, G., D. (1999). *The Context Toolkit: Aiding the Development of Context-Enabled Applications*. In CHI'99 Conference on Human Factors in Computing Systems. Pittsburgh, Pennsylvania, USA.

Schilit, B. and Theimer, M. (1994). *Disseminating Active Map Information to Mobile Hosts*. IEEE Network, 8(5), 22–32.

Schmidt, A., Aidoo, K., A., Takaluoma, A., Tuomela, U., Laerhoven K., V., and Velde, W., V. (1999). *Advanced Interaction in Context*. In HUC'99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, pp. 89-101, London, UK.

Schmidt, A., Beigl, M. and Gellersen, H., W. (1999). *There is more to context than location*. Computers and Graphics Journal, 23(6), 893–902.

SpringSource (2007). *http://springsource.org/*.