Towards Ubiquitous Services Design and Development Approach

Aicha Azoui and Djilali Idoughi(✉)

Applied Mathematics Laboratory – LMA, University Abderrahmane Mira of Bejaia, Bejaia, Algeria
aicha.azoui@gmail.com, djilali.idoughi@univ-bejaia.dz

Abstract. Today's evolution of mobile technologies, telecommunication infrastructures and service oriented paradigm is leading to the development of ubiquitous services. Ubiquitous services are software applications that have the capability to run anytime, anywhere and on any device with minimal or no user attention. However, the advancements and diversity in technologies, the dynamic and ubiquitous nature of these services increase the complexity of the underlying development process. In this paper, we propose a design approach for developing service oriented ubiquitous systems considering both business and ubiquitous requirements. The approach is applied to a crisis management case study.

Keywords: Ubiquitous system · Service orientation · Ubiquitous service · Ubiquitous requirements · Design framework

1 Introduction

The Ubiquitous Computing or UbiComp [1] is currently used to describe a computing environment in which computation is everywhere and computer functions are embedded and connected with various entities so that anyone can access, communicate, exchange and share information anywhere and anytime.

Broadly defined, the vision of ubiquitous computing environment considers the environment which surrounds the user's everyday lives and activities, saturated with computing devices and communication capabilities, which can be implemented in a varied scale of spaces where information and services are provided to the users when and where desired [2].

Figure 1 illustrates broadly the vision of ubiquitous computing environment. The convergence of mobile technologies in terms of mobile devices and telecommunication infrastructures and software engineering paradigms (i.e., especially the emergence of service oriented paradigm (SOC)) has brought about a new service-oriented computing paradigm known as ubiquitous services or context-aware services [3, 4].

However, these advancements and diversity in technologies and the highly dynamic nature of ubiquitous services contribute to make ubiquitous services design a complex and challenging software engineering task compared to conventional services. Thus, the design and the development of ubiquitous services is a more complex task [3, 6]. In conventional services input information can be managed in a common way since it is

N. Streitz and P. Markopoulos (Eds.): DAPI 2016, LNCS 9749, pp. 3–14, 2016.
DOI: 10.1007/978-3-319-39862-4_1

Fig. 1. Global ubiquitous computing environment vision [5]

mainly considered to be obtained principally from the user. In contrast, when developing ubiquitous services, input information needs to be managed differently since it is obtained from diverse sources [7]. This information, termed "context", affects both the behavior and the interaction of the service with the user in a variety of different ways, and enables the service to adapt its functionality accordingly.

In this paper, we aim to present an approach to develop ubiquitous systems based on the service-oriented paradigm providing the end-user with ubiquitous services that meet their current needs, and that adapt to their contexts.

The Remainder of this paper is organized as follows: Sect. 2 presents some related work to the development of service oriented ubiquitous systems. Section 3 presents the basic concepts of the proposed framework which is applied to a typical crisis management case study in Sect. 4. And finally, we conclude and give some further research work in the Sect. 5.

2 Related Work

The development of ubicomp services based systems has led to the emergence of different research works that have provided mainly design frameworks and systems architectures. The most proposed works concentrate mainly on one specific service design issue such as, discovery [8], composition [9] or adaptation [10]. In Chaari *et al.* [10] the focus is on the services adaptation to different contexts in a ubiquitous environment. This adaptation is achieved with a system containing management and storage of context and adaptation modules. Toninelli *et al.* [8], in a ubiquitous scenario, provide a middleware which adopts semantic techniques to perform the discovery of context-aware services based on the requirements and preferences expressed by mobile users. The middleware exploits the users' devices and service profiles metadata to provide personalized and customized services. Meanwhile, Tili *et al.* [9] propose an architecture

model for the service composition which is based on an assembly of lightweight components. The model relies on a software and hardware execution environment evolving dynamically. It is based on a web service for the device infrastructure using events, and dynamically discoverable in a distributed way.

Moreover, some other research work [7, 11, 12, 4, 13] suggest the employment of a model driven engineering process which is an approach for using models at various levels of abstraction in software development. 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. Therefore, Achilleos *et al*. [7] propose a model-driven development process that facilitates the creation of a context modeling framework simplifying the design and implementation of ubiquitous services. In this process, functional and ubiquitous requirements analyses are lacking and adaptation mechanism of the created services is not addressed. Vale and Hammoudi [11] propose the use of the model driven engineering approach to develop context-aware services by the separation of concerns in different models. However, the requirements analysis is not taken into consideration, as well as the proposed context meta-model is not generic and the adaptation strategy is not described. Sheng and Benatallah in [12] present a UML based modeling language for the model driven development of context-aware services and a generic meta-model of context is presented. The proposed approach does not provide guides and instructions for context modeling and requirements analysis. Moreover, the authors do not specify the mechanism used to fulfill context aware services adaptation. In [4] the authors propose an architecture for the development of context-aware services. The development work is based also on the aspect oriented paradigm. The aim of this architecture is to address fundamental challenges for the design of services in context-aware service-oriented systems. This architecture allows the creation of generic context meta-models and adaptation strategy. However, the functional specification of the system and identification of services are not addressed. Moreover, it does not provide guides for identifying context information. Finally, Abeywickrama and Ramakrishnan [13] propose an engineering approach for context-aware services that models and verifies these services at the architectural level. The approach benefits from several software engineering principles such as the model driven architecture, the separation of concerns through aspect-oriented modeling and formal verification using model checking to facilitate context aware-services engineering. The approach development process is incomplete because it does not take into account the analysis phase and a context model is not given.

The most approaches highlighted above have focused on the context modeling and adaptation to context changing as dominant and unique aspects of ubiquitous systems. However, the functional aspect of the system is neglected, this makes these approaches incomplete. In addition, the requirements analysis phase identifying business and ubiquitous requirements yielding to necessary services and context models is absent in all the approaches presented. Therefore, the aim of this paper is to present a service based ubicomp system design framework which overcomes these lacks and provides a richer engineering process as it is described in the next section.

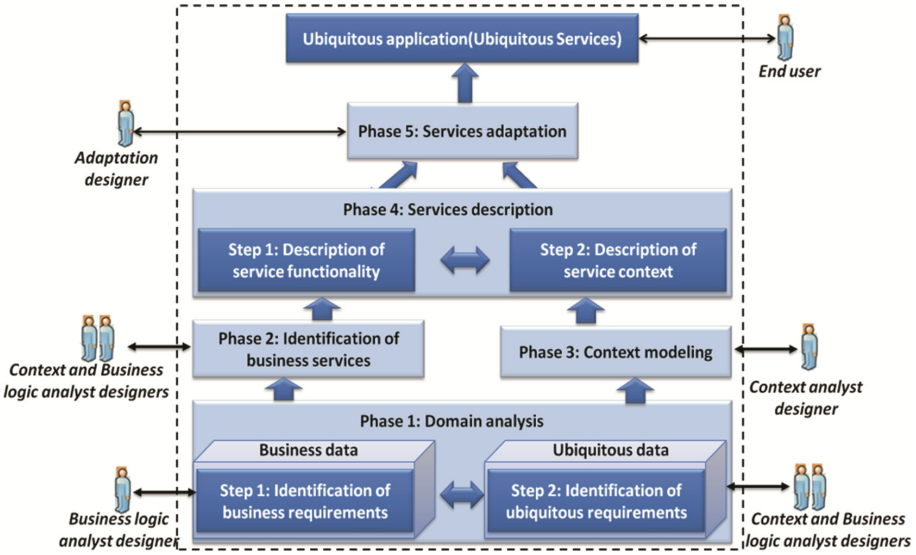

Fig. 2. The global view of the design framework along with the involved design roles

3 Our Proposal

3.1 The Framework Basic Concepts

The proposed framework is based on a set of founding concepts and principles such as facilitating separation of the developer's tasks and specification of ubiquitous models. It considers ubiquitous services as a central concept resulting from building both service oriented business and ubiquitous views. Thus, we separate the development process into business and ubiquitous views. The business view focuses on the business logic of the system and its functionalities. Whereas, the ubiquitous view is responsible for the definition of a context, acquisition, interpretation, modeling and exploitation by the application. Moreover, four different actors (roles) may be involved in the development process of ubiquitous services (Fig. 2) as follows:

1. **The Business logic analyst designer,** focuses on the business logic of the application model without ubiquitous and technological considerations.
2. **The Context analyst designer**, responsible for the identification of ubiquitous functionalities supported by the system. These functionalities involve context data specific to the users and their environment.
3. **The Adaptation designer,** defines a set of mechanisms and rules necessary for the dynamic adaptation of business logic to the context of the application at runtime.
4. **The End user**, the human actor who invokes services anywhere, anytime. The system, transparently, provides the appropriate services using the context.

3.2 The Proposed Framework

The proposed approach is a methodological framework (Fig. 2) which consists of five phases. Each phase includes one or more steps that can be performed by different actors involved (business, ubiquitous). It is an iterative process aimed to achieve user satisfaction and monitor requirements evolution.

Phase 1: Domain analysis. This phase is a preliminary study of the business domain by analyzing both business and ubiquitous issues to elicit and identify business and ubiquitous requirements. It identifies the business processes through different domain scenarios. The ubiquitous requirements are linked to business processes. The outcome of this phase is a domain model and a business use case model enriched with ubiquitous requirements which expresses the potential functionalities of the ubiquitous system.

Step 1: Identification of business requirements. The aim is to identify what the system performs or does; it is performed by the business logic analyst-designer who proceeds to an overall identification of the various business processes described in the functional requirements of the system. As a consequence, a list of business processes relevant to the domain is obtained.

Step 2: Identification of ubiquitous requirements. This step is performed by the context analyst-designer in collaboration with the business logic analyst-designer. It consists in analyzing the business requirements to extract and identify ubiquitous requirements. The context analyst-designer analyzes these requirements and places them in the business scope. The result of this step is a use case diagram model highlighting the overall business processes, their interdependences along with actors and the associated ubiquitous requirements.

Phase 2: Identification of business services. This phase is realized by the business logic analyst-designer collaborating with the context analyst-designer. The purpose is to identify a set of candidate business services from business processes identified in the previous phase. Our approach is based on the concept of goals underlying the business services. Each business process strategic goals are elicited. Thus, the strategic goal is decomposed into one or more functional and sub functional goals. This detailed goals decomposition enables to identify the goals of fine granularity (operational goals) as business services. Afterwards, a goals hierarchical structure is obtained. The realization of this hierarchy takes into account both the ubiquitous and functional goals.

Phase 3: Context modeling. This phase is the first ubiquitous phase. The context analyst designer formalizes the context by analyzing the different ubiquitous goals identified in the previous phase as well as the definition of the context model relative to the ubiquitous functionalities.

Step 1: Definition of ubiquitous rules. The different ubiquitous goals identified in the previous phase should be detailed by transforming them into ubiquitous rules. The generated rules are written in the following form: **If** condition **then** consequence.

Step 2: Generation of the Context model. The set of ubiquitous rules defined in the previous step allows the generation of context data to be considered in the system and to be modeled. The generated conceptual context model includes the entities, their profiles and their interdependences with the environment (Fig. 3). It contains only the needed pertinent elements to model the context and eliminates details that could affect its generic character.

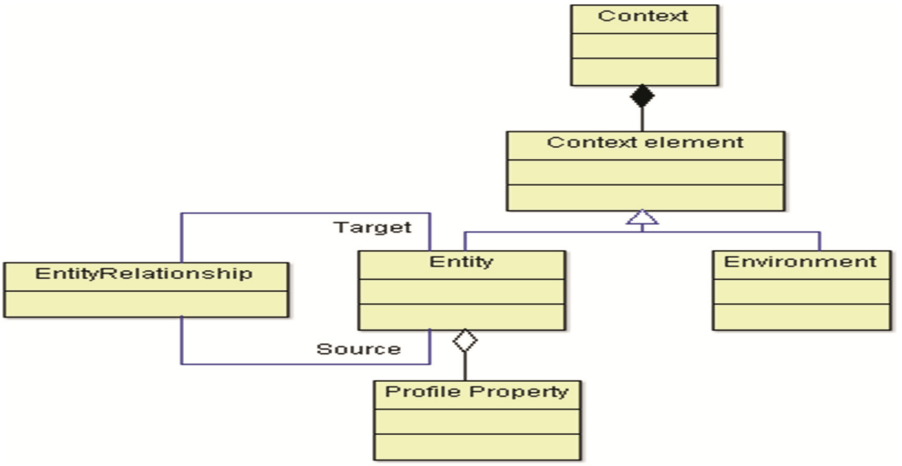

Fig. 3. Context meta-model

Phase 4: Services description. For each identified service, both its functionality that meets user's needs and the context in which the service is valid and executed will be described.

Step1: Description of the service functionality. This description contains information about the functionality that this service can provide. All this information is grouped into an interface.

Step2: Description of the service context. This description contains the information about the context of the service such as the type of device, network, the time required to deliver to the user the service final response, cost of service, etc. This context presents additional information about the service in order to determine whether or not, this service is relevant to the current context of the user, and improve the quality of the response returned to the user.

Phase 5: Services adaptation. The adaptation designer defines a strategy for a dynamic service adaptation to new contexts of use. Aspect-oriented programming [14] is used to adapt the service behavior according to contextual changes, without modifying its business logic. The separation between business services considerations and contextual considerations remains fundamental to the adaptation of services. Therefore, modification of the adaptation actions should not cause changes in the business logic of the

service. These aspects will be dynamically woven into the core service to be adapted (Fig. 4).

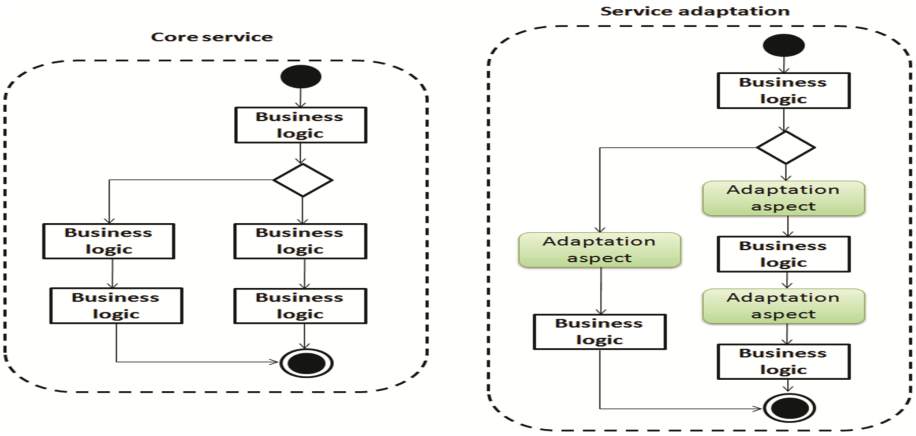

Fig. 4. Service adaptation strategy

The following section describes broadly the application of the framework on a concrete case study relative to the crisis management field.

4 Case Study - Crisis Management Field

4.1 Scenario Description

Crisis management is a special type of human and complex organization [15] in which, various actors belonging to different authorities need to collaborate and work together with the shared aim to solve, or at least reduce, a crisis situation. Each actor may be equipped with different devices and communication technologies to carry on specific tasks. In this paper, a crisis can be defined as a disruption in the normal functioning of an organization or society, resulting from a brutal and sudden event. The main crisis management activities can be grouped into four phases: (1) prevention, (2) preparation, (3) emergency management (response) and (4) recovery.

Hereafter, we focus on the emergency management phase where the underlying response process is divided into two main steps (Fig. 5): (1) Alert and (2) Intervention.

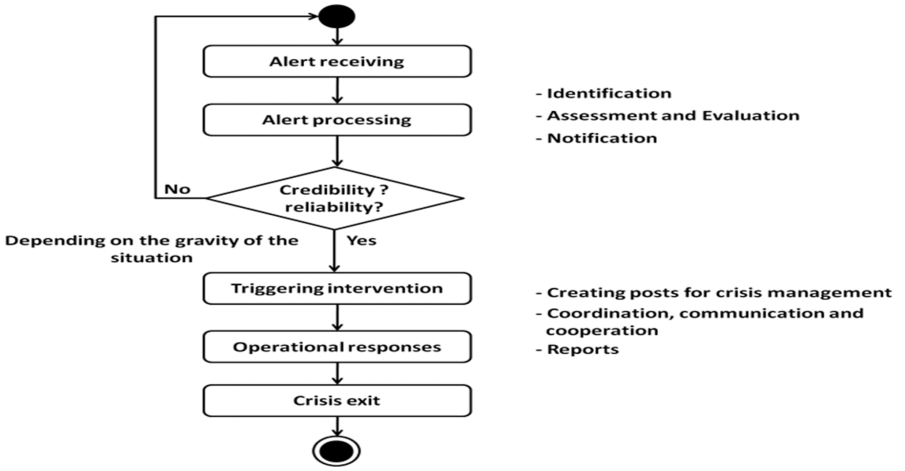

Fig. 5. Emergency management process

The Alert step represents a significant level of emergency that informs required people depending on the scope and gravity of the situation. It is triggered by the detection of an event or upon a receipt of an incident report. Afterwards, an evaluation of relevance of the emergency is determined, taking into account various types of data (affected people number, number of responders required in a short lapse of time, intervention of one or several governmental departments, etc.). Hence, a series of preplanned strategic tasks such as quickly mobilizing resources and required of various concerned organizations are carried out.

4.2 Application of the Framework

Phase 1: Domain analysis. A set of business processes have been identified. Two business processes are identified for emergency management: (1) *Alert Processing* and (2) *Intervention Order*.

Phase 2: Identification of business services. A hierarchy of functional, operational and ubiquitous goals corresponding to "*Alert processing*" process is obtained. The *Alert* process must satisfy the strategic goal "*Improve the time of detection and response"*. The ubiquitous goals "*Consider personal information and context*" and "*Adjust the display*" are directly linked with the strategic goal which means that context information should be considered in all sub-functional and operational goals.

After the decomposition of the *Alert Processing* business process, four business services are identified: *Monitoring*, *Receiving Alert*, *Evaluating Alert*, and *Notifying Alert*.

Phase 3: Context modeling. The ubiquitous goal defined in the hierarchy is used to derive ubiquitous rules to identify the necessary context information which facilitates context modeling. Table 1 illustrates the derived ubiquitous rules that are applied to the *Alert processing* business process.

Table 1. A driven set of ubiquitous rules

Ubiquitous rules	Conditions	Consequences
The location must be considered before sending an alert	Detection of a disturbance	Send an alert
The surrounding disturbances have to be considered during the movement of the rescuers	Damage on infrastructure (roads)	Inform rescuer
	A traffic alert	Inform rescuer
The context of the rescuer (location, device and environment) must be considered before notifying	The crisis is large-scale	Send notification
The display must consider the context of the rescuer and his preferences	Preference of display	Interact by respecting the preferences

Figure 6, illustrates a context model for the crisis response obtained by instantiating the context meta-model (Fig. 3) and considering the set of ubiquitous rules above.

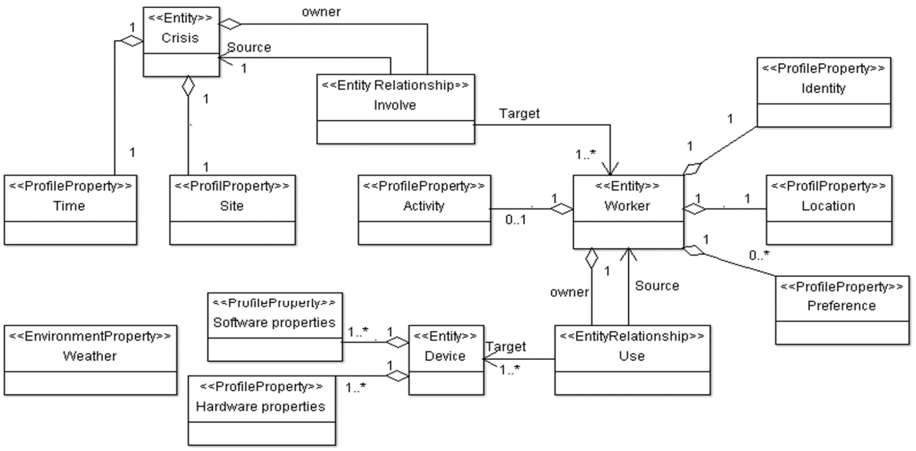

Fig. 6. A context model for crisis response

Phase 4: Services description. The Fig. 7 shows an extract of the WSDL (Web Service description Language) document describing a hardware resource allocation service invoked by an intervener on the crisis site. This service has a list of material resources which are available during the crisis. When this service is invoked by the intervener, the service enables the selection of the desired equipment to validate the allocation in this latter. The WSDL description presents only information about service functionalities. Moreover, we take into account the context of the service. This context is defined by three XML elements: “device”, “service” and “network”.

```
<definitions targetNamespace="http://ws/"
name="ressource_allocation_serviceService">
<types>
    <xsd:schema>
<xsd:import namespace="http://ws/"
schemaLocation="http://localhost:8080/ressource_allocation/ress
ource_allocation_serviceService?xsd=1"/>
    </xsd:schema>
</types>
<message name="Allouerressource"> .... </message>
<message name="AllouerressourceResponse"> .... </message>
<portType name="ressource_allocation_service"> .... </portType>
<binding
name="ressource_allocation_servicePortBinding"type="tns:resso
urce_allocation_service"> .... </binding>
<service name="ressource_allocation_serviceService">
   <port name="ressource_allocation_servicePort"
binding="tns:ressource_allocation_servicePortBinding">
    <soap:address
location="http://localhost:8080/ressource_allocation/ressource_a
llocation_
     serviceService"/>
    </port>
</service>
</definitions>
```

Fig. 7. WSDL document of the service hardware resource allocation

Phase 5: Services adaptation. A crisis management scenario is used to illustrate the service adaptation process. A rescuer on the disaster site makes a resource allocation request for intervention. Information about the user's context is captured either implicitly or explicitly and stored in an XML document. The rescuer may need to interact with the same service in two different contexts. To ensure service dynamic adaptation to the user context, the *services manager* compares the current and former contexts. To achieve this, it contacts the *context manager* by requesting both contexts to check whether they are compatible or not. If not, the *services manager* specifies the list of changes in the current context and requests the services *context manager* to verify if this context is

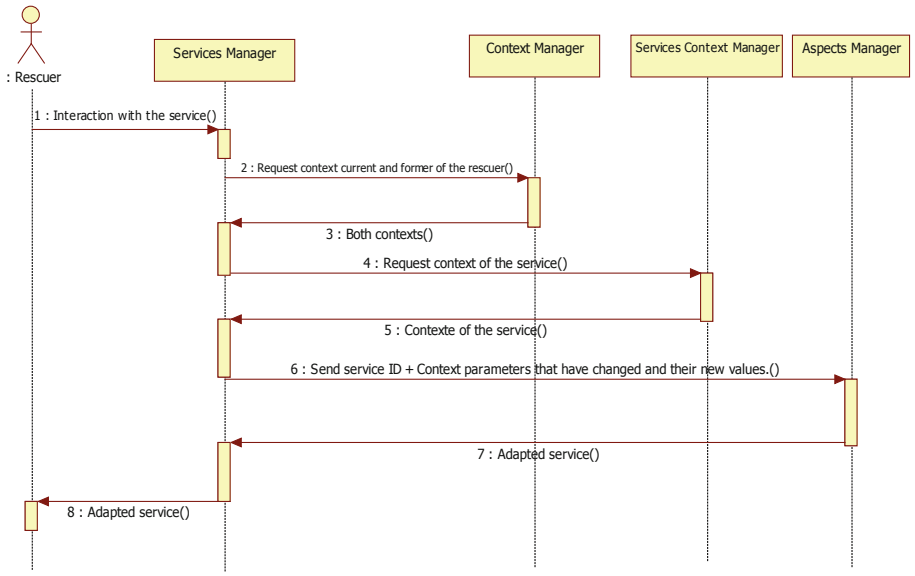

Fig. 8. Sequence diagram for the dynamic adaptation process

compatible with the list of changes in the current context or not. Similarly, it compares the value of each element in this list with the context of the service. If they are not compatible, the *services manager* returns the identifier of the service (service ID) as well as context parameters that have been changed to the *aspects manager*. The *aspects manager* selects appropriate aspect for adapting the service. The selected aspect must contain the adaptation action corresponding to the change of context. Thereafter, this aspect is woven within the core service to be adapted and provides a relevant response to the end user as illustrated by the sequence diagram in Fig. 8.

5 Conclusion

In this paper, a design framework for developing ubiquitous system based on the service-oriented paradigm is proposed. This provides a high level of abstraction and has several advantages. In addition to the benefits of productivity and quality improvement of context and ubiquitous services development, the framework has also the advantage of considering ubiquitous requirements to extract ubiquitous rules and provide formal procedures. This is to identify context information needed to build easily the context model using a predefined context meta-model. The context meta-model is generic and open to allow its extension to various domains depending on needs. Ubiquitous requirements are developed in a separated way from system functionalities (i.e., business requirements), which is useful to enhance flexibility and facilitate system reuse and modifications. The framework is applied on a real case study related to the crisis management field.

Acknowledgments. The authors gratefully acknowledge and express their warm thanks to the Direction Générale de la Protection Civile de la wilaya de Bejaia, Algeria and the University A. Mira of Bejaia.

References

1. Dhingra, V., Arora, A.: Pervasive computing: paradigm for new era computing. In: First International Conference on Emerging Trends in Engineering and Technology, 2008, ICETET 2008, pp. 349–354. IEEE (2008)
2. Weiser, M.: The computer for the 21st century. Sci. Am. **165**(3), 94–104 (1991)
3. Abeywickrama, D.B.: Pervasive services engineering for SOAs. In: ICSOC Ph.D. Symposium (2008)
4. Hafiddi, H., Baidouri, H., Nassar, M., Kriouile, A.: Context-awareness for service oriented systems. Int. J. Comput. Sci. Issues (IJCSI) **9**(5), 95–104 (2012)
5. Idoughi, D., Azoui, A.: SOA based ubiquitous computing system design framework. In: Proceedings of the 12th ACM International Symposium on Mobility Management and Wireless Access, pp. 71–75 (2014)
6. Preuveneers, D., Berbers, Y.: Semantic and syntactic modeling of component-based services for context-aware pervasive systems using OWL-s. In: First International Workshop on Managing Context Information in Mobile and Pervasive Environments, pp. 30–39 (2005)

7. Achilleos, A., Yang, K., Georgalas, N.: Context modelling and a context aware framework for pervasive service creation: a model driven approach. Pervasive Mobile Comput. **6**, 281–296 (2010). Elsevier
8. Toninelli, A., Corradi, A., Montanari, R.: Semantic-based discovery to support mobile context-aware service access. Comput. Commun. **31**(5), 935–949 (2008)
9. Tigli, J.Y., Lavirotte, S., Rey, G., Hourdin, V., Riveill, M.: Lightweight service oriented architecture for pervasive computing. Int. J. Comput. Sci. Issues (IJCSI) **7**(4), 1–9 (2010)
10. Chaari, T., La forest, F., Celentano, A.: Adaptation in context-aware pervasive information systems: the SECAS project. Int. J. Pervasive Comput. Commun. **3**(4), 400–425 (2007)
11. Vale, S., Hammoudi, S.: Model driven development of context-aware service oriented architecture. In: International Conference on Computational Science and Engineering - Workshops (2008)
12. Sheng, Q.Z., Benatallah, B.: ContextUML: a UML based modeling language for model-driven development of context-aware web services. In: 4th International Conference on Mobile Business (ICMB 2005), pp. 206–212 (2005)
13. Abeywickrama, D.B., Ramakrishnan, S.: Context-aware services engineering: models, transformations, and verification. ACM Trans. Internet Technol. (TOIT) **11**(3), 10 (2012)
14. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C.V., Loingtier, J.M., Irwin, J.: Aspect-oriented programming. In: Akşit, M., Matsuoka, S. (eds.) Object-Oriented Programming, ECOOP 1997. Lecture Notes in Computer Science, LNCS, vol. 1241, pp. 220–242. Springer, Heidelberg (1997)
15. Aitabdelouhab, K., Idoughi, D., Kolski, C.: Agile & user centric SOA based service design framework applied in disaster management. In: ICT-DM 2014, 1st IEEE International Conference on Information and Communication Technologies for Disaster Management (2014)