Towards A UML Profile for Context-Awareness Domain

Mohamed-Salah Benselim[1] and Hassina Seridi-Bouchelaghem[2]
[1]Department of Management Science, University of "08 Mai 45", Algeria
[2]Department of Computer Science, University of Badji Mokhtar, Algeria

Abstract: *Defining Unified Modelling Language (UML) profiles allows adaptation of the UML metamodel for specific domain, area, platform, etc. Context awareness is one of particular domains that need to be well adapted when we use UML language to model specific situations of users and applications. Therefore, it is necessary to create specific modelling notations for this particular domain. In this paper, we present an extension of the UML notations as a profile used for context-aware applications development in ubiquitous computing environment. The proposed UML context-aware profile is a package of specific profiles that extend the standard notations of three UML diagrams chosen according to different views of a system (use case diagram, sequence diagram and activity diagram). For each diagram, we propose UML extension mechanisms such as stereotypes, constraints and tagged values that can model any contextual situation by an adequate graphic representation. Each element of the context of use should be able to be represented by this UML profile. To demonstrate the feasibility of our work, an example in medical field is shown by using StarUML software modelling platform. This work will complete the list of extended notations (class diagram) presented in previous work in order to propose a more complete UML profile.*

Keywords: *Software engineering, ubiquitous computing, UML, profile, extension, context-aware, modelling, metamodelling.*

Received February 10, 2014; accepted December 23, 2014

1. Introduction

Using modern information systems must take into account several features of pervasive environment such as user mobility, information heterogeneity and systems distribution. These features can provide precise indications concerning a contextual situation and identify the context of use of this situation. We note that in all our study, we take as reference the definition of the concept of "context" cited in [7]. A contextual situation is a part of the process of executing a system by a user at one defined time. It is to say that a user can be in different situations when using a system because he is influenced by the change of factors (or contextual elements) like: time, location, used device, nearby persons, …, etc. So, each situation is defined by a set of variable values of these elements and that constitute the context of use of this situation. For this, we use the term of "contextual situation" to define the variable situation cases of the context of use. In pervasive environment, a user can change, at any time, one or more contextual elements. For example, he changes his location from office to home or he changes the device he used from cell phone to personal computer. This user has transited from one contextual situation to another contextual situation because of changing the context of use. Here, the system (or application) must be well adapted to provide relevant information according to each situation. For covering context changes, specific modelling notations are needed in context-aware applications development. In previous works, [4, 6], we proposed extended Unified Modelling Language (UML) notations destined for modelling the context of use with only one UML diagram that is class diagram. These proposed notations represent a part of an UML context-aware profile that we present in this paper. Our proposal aims to complete these works by providing new modelling notations for UML diagrams (other than class diagram) and that are regrouped in a profile especially for the development of context-aware applications. This profile is structured as a package of three profiles that comprise extended notations of three UML diagrams (use case, activity and sequence. The proposed UML profile allows adaptation of the UML metamodel for context awareness domain. This profile is defined by a set of extensibility mechanisms such as stereotypes, constraints and tagged values that are applied onto different UML diagrams. These diagrams are chosen according to different views of a system. The functional view is represented by a use case diagram and the static view is represented by a class diagram. Also, the dynamic view can be represented by a sequence diagram or by an activity diagram. That is to say that use case diagram shows the system functionalities and class diagram represents the system structure. Dynamic sequencing of system tasks is modelled by a sequence diagram or an activity diagram. The remainder of this document is organized as follows: section 2 summarizes several previous works related to the same research area in order to position our problematic. Section 3 presents an

overview of our study problematic, motivations and goals to reach. In section 4, we expose details of the proposed UML profile while showing extensibility mechanisms for three UML diagrams: Use case, activity and sequence. Section 5 shows implementation of the used concepts in a chosen platform and a running example is presented in section 6. Finally, in section 7, we conclude this work and we present perspectives for future research.

2. Related Works

In this section we present several previous works that are related to our study. First, we list works that use the notion of "profile" in different specific domains of computing area. Then, we show the recent works that introduce profiles in context awareness domain. At the end, we try to position our study according to the listed works. If we see the usage of UML as a modelling language, there is a large number of works that are proposing specific profiles for specific domains such as for the description of software architecture [2], the Web Ontology Language (OWL) ontologies [8], multidimensional modelling in data warehouses [22], business process modelling [20] and [16], software product lines [33], aspect-oriented software development [1], software architecture descriptions [18], mobile systems [11], communicating systems [32], real-time systems [10], object oriented expert system [27], agent-oriented modelling [31], service-oriented architecture [12], platform independent modelling of service-oriented architecture [21], description of dynamic software architecture [17], developing airworthiness-compliant [34]. Other works have investigated the domain of context-awareness in different ways such as cited below. Hsu [15] proposed "Web2.0MUML" Model Driven Architecture (MDA) approach and UML profile for modelling "mashups" (specific Web 2.0 technology). The Web2.0MUML profile extends UML by using a profile mechanism for Web 2.0 mashup modelling that presents the relevant structural properties of Web 2.0 at the conceptual level. In other study, Hsu [14] presented a UML profile for modelling Web 2.0 based context-aware applications. Also, Hsu developed a multi-layer context framework that integrates Web 2.0 technologies into context-aware system for supporting ubiquitous mobile environment [13]. This framework includes context sensor layer, context information layer, context service layer, context representation layer, mobile device layer, and context-aware mobile Web 2.0 application layer. Fuentes *et al*. [9] proposed the use of aspect-oriented modelling UML profile for designing pervasive applications. This technique contributes to the encapsulation of crosscutting concerns (context-awareness) into well-localized modules. Korherr and List [19] extend the UML 2 activity diagram to make process goals and performance measures conceptually visible. Also, they provide a mapping to Business Process Execution Language (BPEL) to make the measures available for execution and monitoring. The work of [25] uses models to develop context-aware web services. It proposes a metamodel that permits modelling any contextual information and taking into account the separation of aspects and the context in the early stages of development. Van and Coninx [30] presents a more detailed analysis describing how UML and its light-weight extension mechanism (as profiles) can be used to model both the dynamic and structural aspects of context-sensitive user interfaces; and for this, an approach has been presented to create context-sensitive user interfaces using models expressed in UML.

Our study seeks to complete these works by proposing an UML context-aware profile for the development of context-aware applications. This profile provides more concepts and notations expressed in explicit way and designed for modelling contextual information. The proposal in this paper is a continuation of our previous works in the same field research. Indeed, we introduced a new vision of MDA approach to develop context-aware application in ubiquitous information systems [5]; then we presented an overview of modelling context with extended UML in [3, 6]. In the last paper [4], we began to treat UML diagrams with details and we focused on class diagram notations. Our actual proposed profile has to facilitate the development of context-aware applications because it provides the main modelling tools needed for a development process. In this paper, we present new extended notations for three UML diagrams (use case diagram, activity diagram and sequence diagram).

3. Motivations and Objectives

A model is an abstraction of a system that provides an easier way for the development of complex applications. Models are used to simplify the system views modelling. MDA is an OMG approach [24] that is based on model and model transformation. UML is the most used language with different paradigms and domains. UML is a standard object oriented modelling language offered by the Object Management Group (OMG) and it is used for modelling many aspects of software systems [28, 29]. UML proposes general notations that can be used for the system development process in different areas and domains. In other words, UML allows modelling all domains but with general and standard concepts and notations. The particularity "standard or general" of the UML language has to limit its opportunities while working in particular domains that need specialized notations. Seeing the works presented in the previous section, we note that UML has not specific notations for representing context information. So, the standard UML concepts do not support all aspects of the context of use in an adequate

manner. Therefore, we decided to find a better solution to represent the context with explicit and appropriate notation and by creating a new UML profile especially for context-awareness domain. Now, we try to explain the statement of our problematic and answer the following questions:

- Why we are obliged to extend UML for context-awareness domain?
- Can UML be profiled to support context-aware modelling?
- What we are going to really add to the existing standard UML?
- How to implement the extended notations to build an UML profile?

Information processing tasks, especially in the context-awareness domain, are becoming more difficult and complex. Ubiquitous computing is an emerging field research in computing domain that assures information processing independently of time, space, device, …, etc. It permits to make available all needed information everywhere and anywhere. Each application should be able to adapt its services with the change of each contextual situation and satisfy all users' specific preferences. In ubiquitous computing, we must take into account changing features of a system such as user's mobility, information heterogeneity and systems distribution. These features represent the context of use of a situation and their common specificity is the continuous changing because we are working in ubiquitous environment and all of the system actors and components can change in time and space. For example, the modelling of "user" entity cannot be limited as a static entity (as in class diagram) or as a simple actor (as in use case diagram) such as in standard UML, but it must be represented with an appropriate notation that can show and can take in charge the eventual mobility of this user. In this case, we must think to consider specific features of each new contextual situation such as the state of the user (sitting, standing or walking), the nearby persons (alone or with other persons), the available resources, the used device and the existing networks. Here, we note that standard UML does not propose explicit notations to model contextual features, and thus, it cannot be well used for the development of context-aware applications in ubiquitous environment. UML is a reference metamodel that can be extended or customized to be adaptable to a particular domain such as context-awareness domain. The extension process consists on adding new notations to the existing ones by creating extensibility mechanisms including stereotypes, constraints and tagged values. These new notations allow the customization of both syntax and semantics of the standard UML elements. The union of these extensibility mechanisms permits to construct a profile that allows adaptation of the UML metamodel to model context-aware application with appropriate notations and concepts. The new notations will be able to express specific concepts of particular domains.

Our goal is to expand the standard UML by extending a part of its metaclasses in order to build an UML context-aware profile more adapted for modelling process in ubiquitous environment. The proposed profile will contain stereotypes, constraints and tagged values. Stereotypes permit to extend the UML notations in order to create new model elements and they define how an existing metaclass may be extended in the profile. Constraints are associated to stereotypes in order to make restrictions or conditions on the corresponding metamodel elements. Tagged values are considered as the values of stereotype properties. When an extension mechanism is applied to a model element of UML metamodel, a new element notation (or concept) is derived from this model element according to the requirements of the context-awareness domain. For example, we can derive (or extend) the UML metaclass "Object" to construct the stereotype <<ContextObject>> that will permit to model contextual entities (entities which are comprised in a contextual situation) of an UML sequence diagram. Also, we can use the stereotype to model specific elements of our domain such as a "ContextActor". The "ContextActor" class is a stereotype which can be derived from the standard UML metaclass "Actor" by extension. This stereotype is specialized in order to represent all nomad and mobile users of the system such as (in health domain): patient, doctor, nurse, pharmacist, student of medicine faculty, professor of medicine faculty etc., Then, constraints (restrictions or conditions) are attached to each of obtained stereotypes in order to limit and to guide the use of this new element according to the specific characteristics of the system. The built of a stereotype is completed by creating tagged values (attributes of the stereotype). Each of these tagged values is defined by specifying a property name and its value in order to provide more opportunities for modelling contextual information. After we do the same operation to all specific characteristics (entities, relationships, behaviour, etc.) of our particular domain, we group all obtained concepts in a package which will be the UML profile for the development of context-aware applications in ubiquitous environment.

This profile is feasible by implementing its concepts and notations with StarUML software modelling platform. StarUML is an extensible platform which supports UML language and provides excellent extensibility, customizability and flexibility [26]. Stereotypes are defined in the document file of the profile by using eXtended Markup Language (XML) format. For each stereotype we indicate the name, a short description and the base UML class. Constraints are introduced by using the "Constraint Editor" of StarUML menu System. For each constraint, we can specify its body by using the natural language or using

Object Constraint Language (OCL) [23]. The "Tagged Value Editor" of StarUML menu System can be personalized with needed properties and by specifying these properties and their values with an XML document. Then, the StarUML system menu is customized by adding an entry option in order to facilitate the use of the proposed profile from the StarUML main menu. Each developer can use the proposed concepts and notations by including the new profile in the StarUML "profile manager editor". This operation will allow loading all the profile components to be used by the application developer. In this section, we aimed to motivate and to argue the need of a specific UML profile in the ubiquitous computing environment. To explain the problematic of this work, we tried to answer three main questions (what?, why? and how?) summarized as follows:

? Question 1: What?
✓ Answer 1: Creating specific UML profile for a specific domain (context-awareness)
? Question 2: Why?
✓ Answer 2: Standard UML does not propose (in explicit way) modelling tools of the context of use
? Question 3: How?
✓ Answer 3: By extending the existing UML elements to support context modelling.

Our proposal will provide several benefits of use such as: Augmenting software productivity, optimizing development effort, minimizing time and decreasing costs. This is able to make the context-aware application development easier and more efficient. Briefly, the main goal of this study is the creation of specific notations to model specific information, and thus, the creation of a specific UML profile for a specific domain (context-aware computing).

4. UML Context-Aware Profile

In this section, we expose details of the proposed UML context-aware profile by showing its structure and components. These details concern the description of the profile package, creating stereotypes, creating constraints and creating tagged values; and this, for three UML diagrams (use case, activity and sequence). It is necessary to note (gray colour) in Figure 1 and in Table 1, that extended notations of UML class diagram have already been represented in previous work [4]; and thus, they will not appear in this paper. Also, we note that we are going to respect concept alias used in that work because it is a continuation of the same project.

4.1. UML Context-Aware Profile Package

The context of use can be represented by specific UML notation. Thus, we have to extend the unified modelling language by adding new elements that can represent the contextual characteristics in appropriate form. Each of these characteristics must be represented by an adequate notation of UML language. For this, we propose a standard UML extension as a profile that contains stereotypes, tagged values and constraints. A stereotype permits to define a new meaning of an existing UML metamodel element. Tagged values are always attached to a stereotype and their role is to indicate attributes of the created stereotype. Constraints define the restrictions of semantics for each added new element. Our proposed profile is a structure diagram which describes lightweight extension mechanism to the UML language by defining custom stereotypes, tagged values and constraints. This profile can be represented by a package of other profiles that extend UML metamodel as shown in Figure 1. At this stage of the survey we chose three UML diagrams (use case, sequence and activity) that represent the main tools of an application development (analysis, design, and implementation). New UML diagrams notations are obtained by extending the existing elements of UML metamodel.

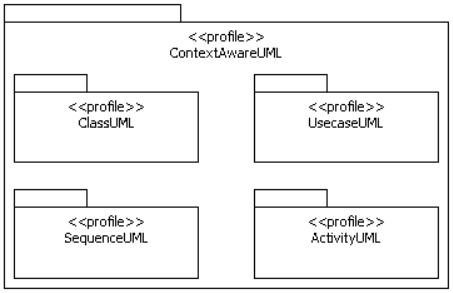

Figure 1. Architecture of the proposed profile package.

The UML context-aware profile package is composed of four major profile packages: ClassUML [4], UsecaseUML, SequenceUML and ActivityUML. Each package is inherited from the common UML metaclass 'package' and will comprise extended notations of the correspondent diagram as shown in Table 1.

Table 1. Description of included profiles of the proposed profile.

Profile Name	Reference Metamodel	Metamodel Element (Metaclass)	Description
ClassUML Profile	UML Metamodel	'Package'	Extends UML class diagram notations to support context-aware modeling
Usecase UML Profile	UML Metamodel	'Package'	Extends UML use case diagram notations to support context-aware modeling
SequenceUML Profile	UML Metamodel	'Package'	Extends UML sequence diagram notations to support context-aware modeling
ActivityUML Profile	UML Metamodel	'Package'	Extends UML activity diagram notations to support context-aware modeling

In Table 2, we list the main UML metamodel elements that will be extended and studied in this paper.

Table 2. UML diagrams and correspondent extension mechanisms.

UML Diagram Name	Stereotypes	Metamodel Element (Metaclass)
Use Case Diagram	ContextActor	Actor
	ContextUseCase	Use case
	ContextDependency	Relationships (use, extend, include)
sequence Diagram	ContextObject	Object
	ContextLifeLine	LifeLine
	ContextMessage	Message
Activity Diagram	ContextActivity	Activity
	ContextTransition	Transition

Proposed UML extension enables an appropriate use of the domain specific notation in place of the standard UML concepts. System entities which have specific properties and constraints that cannot be represented with standard UML notations have to be modelled with the proposed stereotypes. The specific characteristics (limits, conditions, mean, semantics, ...) will be taken in charge in stereotypes by attached constraints and tagged values. Below, we are going to describe the three profile packages and we expose the main proposed concepts and notations to respectively three UML diagrams (use case, activity and sequence). It is necessary to note that, in MDA architecture, defining a UML profile means that standard UML will be extended at the metamodel level M2. Thus, proposed extensions are done at metamodel level M2 and then diagram notations (at model level M1) are built to be conform to these metamodel concepts.

In use case diagram, existing UML notations will be extended in order to obtain new notations that will be more adapted to model specific situations and scenarios in context-aware domain. The stereotype <<ContextActor>> is obtained from the standard UML metaclass "Actor" by customizing its syntax and semantics. This customization permits to limit the field of use of "Actor" concept to be used in modelling specific users and entities such as nomad users, travellers, distributed systems, mobile devices, etc. In this part of our proposed profile, we extend the main metaclasses of standard UML use case diagram to create new modelling concepts as stereotypes. Here, metaclasses "Actor", "UseCase" and "Dependency" are respectively extended by the stereotypes <<ContextActor>>, <<ContextUseCase>> and <<ContextDependency>> as shown in Figure 2. The difference between old and new concepts is that we attach to new ones (stereotypes) constraints and tagged values. Constraints are used to specify limits and restrictions according to particular specifications of the context of use. Tagged values show the new properties (attributes) of each stereotype.

Generally, all UML concepts can be stereotyped to be adequate with particular domain or platform. Standard UML diagram concepts have to be extended in order to create specific notations (as stereotypes) for the development of context-aware applications in ubiquitous environment.

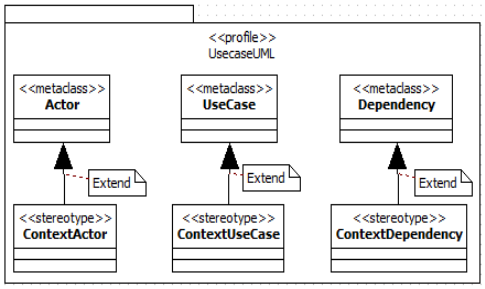

Figure 2. Package of use case diagram profile.

In UML activity diagram, two main metaclasses ("Activity" and "Transition") are extended by two stereotypes (<<ContextActivity>> and <<ContextTransition>>) as shown in Figure 3. In ubiquitous environment and mobile systems, tasks (or activities) perform and execute contextual actions that are influenced by a changing context of use (mobility of users, used devices, heterogeneity of information sources and system distribution). Such activities cannot be represented by standard UML notation because each of them can have several variable forms according to incurred variations of the context of use. Thus, we have to use the proposed stereotypes to model contextual activities.

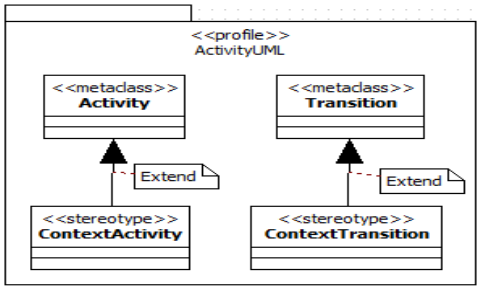

Figure 3. Package of activity diagram profile.

UML sequence diagram allows representing the successive interactions of a use case (or a scenario) in chronological order. It illustrates objects interactions by showing sent and received messages between these objects. Because objects are instances of classes and because we operate in context-awareness domain (for which we have proposed the use of contextual classes such as <<ContextClass>> stereotypes), thus we are obliged to work with <<ContextObject>> concepts as instances of <<ContextClass>> stereotypes.

Figure 4 shows how UML metaclasses ("Object", "LifeLine" and "Message") are extended by new stereotypes (respectively <<ContextObject>>, <<ContextLifeLine>> and <<ContextMessage>>). The created stereotypes provide new meaning and well defined semantics to the existing UML metaclasses by specifying the exact goal to which they will be used. New stereotypes are able to model any object that varies with the context of use of a changing situation.

Object varying characteristics are taken in charge by constraints and tagged values that are attached to the corresponding object stereotype <<ContextObject>>. This will permit customization of the use of UML standard sequence diagram notation in the specific context-aware domain. Also, it will permit, precise and explicit modelling way of contextual factors that are continuously changing.

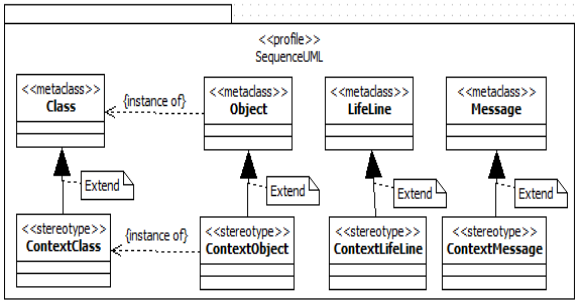

Figure 4. Package of sequence diagram profile.

4.2. UML Context-Aware Profile Stereotypes

The main standard UML metaclasses ("Actor", "UseCase", "Transition", "Activity", "Object", etc.,) are extended in order to create stereotypes that model the different views of a system. These stereotypes define how existing metaclasses may be extended as a part of the proposed profile. A short description of proposed stereotypes is shown in Table 3. For example, in use case diagrams, the <<ContextActor>> stereotype will be used to model entities (such as users, systems, hardware, software, etc.,) that present contextual constraints and that have changing properties. <<ContextUseCase>> is a stereotype that represents the needed functions (composed by contextual actions and varying tasks in time and space) of a system. <<ContextDependency>> shows a specific relationship that relates two <<ContextUseCase>> stereotypes. To appropriately model a nomad user with use case diagram, we need a specific notation that represents explicitly the particular properties of this nomad user such as the user's state (sitting, standing, sleeping, walking, travelling, etc.,), the actual location (home, office, hotel, airport, etc.,), the used device (PC, laptop, mobile, PDA, etc.,), the nearby persons, the available resources, existing networks, etc. These properties will be considered by inserting them as attached tagged values (attributes) to the stereotype <<ContextActor>>.

When the stereotype is applied to a model element, an instance of this stereotype is associated to an instance of the corresponding metaclass. This stereotype will be specialized in several instances of users according to the variation of attached tagged values. In our example, we can model a nomad user as shown in Figure 5.

Table 3. Description of the proposed UML stereotypes.

UML Diagram Name	Stereotype	UML Construct (Metaclass)	Description
Use Case Diagram	ContextActor	Actor	Represents roles that are played by nomad users or other mobile systems and subjects
	ContextUseCase	UseCase	Specifies a set of contextual actions or tasks that are performed by a system
	ContextDependency	Dependency	Is a relationship which indicates that a model element needs another model element (*ContextActor* or *ContextUseCase*)
Activity Diagram	ContextActivity	Activity	Is a set of contextual actions (or methods) corresponding to operations on *ContextClass*
	ContextTransition	Transition	Indicates the transition from the end of a *ContextActivity* to the beginning of another *ContextActivity*
Sequence Diagram	ContextObject	Object	Represents an individual participant (*ContextObject* or its instance) in the Interaction
	ContextLifeLine	LifeLine	Is associated to a *ContextObject* and represents the period of life (as a line) for which this *ContextObject* can participate in the interaction
	ContextMessage	Message	Defines a particular communication between two *ContextLifelines* corresponding to two interaction *ContextObjects*.

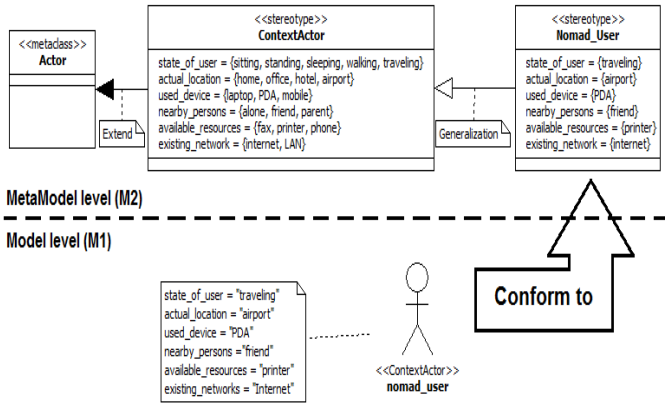

Figure 5. Example of modelling a specific user with proposed notations.

4.3. UML Context-Aware Profile Constraints

UML diagram constraints are defined by clear conditions and restrictions that are specified to limit and to determine how proposed stereotypes (<<ContextActor>>, <<ContextUseCase>>, <<ContextObject>>, <<ContextActivity>>, <<ContextTransition>>, etc.,) are used in modelling process. These constraints show the main differences between proposed stereotypes and existing concepts of standard UML. Generally, they can be expressed or written by several ways and tools such as:

- Natural languages (English, French, etc.,).
- Programming languages (Java, etc.,).
- Mathematical Notations (AND, OR, +, =, etc.,).
- Graphical diagrams (UML diagrams, etc.,).
- OCL language.

Below we try to describe the proposed constraints with three representation kinds (natural language, UML class diagram and OCL language).

Firstly, these constraints are described in Table 4 by using natural language (English). Secondly, we can use UML class diagram to express these proposed constraints such as illustrated in Figures 6, 7 and 8 that

concern respectively use case diagram, activity diagram and sequence diagram.

Table 4. Description of the proposed UML constraints with natural language.

UML Diagram Name	Stereotype	Description of Attached Constraints
Use Case Diagram	ContextActor	It must be extended from the UML metaclass "Actor" It must be related to *ContextUseCase* by *ContextDependency* relationship
	ContextUseCase	It must be extended from the UML metaclass "UseCase" It must be related to another *ContextUseCase* by *ContextDependency* relationship
	ContextDependency	It must be extended from the UML metaclass "Dependency" It relates two *ContextUseCase* It shows the kind of existing dependence (use, extend or include) between related *ContextUseCase*
Activity Diagram	ContextActivity	It must be extended from the UML metaclass "Activity" It must be related to another *ContextActivity* by *ContextTransition* relationship
	ContextTransition	Is a relationship that relates two *ContextActivity* Two attributes must not have the same name Two operations must not have the same name
Sequence Diagram	ContextObject	It must be extended from the UML metaclass "Object" It can interact with another *ContextObject* by *ContextMessage*
	ContextLifeLine	It must be extended from the UML metaclass "LifeLine" It supports the points (where and when) of receiving and sending *ContextMessage*
	ContextMessage	It must be extended from the UML metaclass "Message" It must start from a *ContextLifeLine* (of a source *ContextObject*) to arrive to another *ContextLifeLine* (of a target *ContextObject*)

Thus, in Figure 6, we present the proposed use case diagram constraints by specifying the type of relationships between stereotypes and by fixing corresponding multiplicities. By applying these constraints, we fix the number of <<ContextUseCase>> instances that can be related by one <<ContextDependency>> instance (use, include or extend) to '2' and we authorize that a same <<ContextActor>> instance can trigger one or more <<ContextUseCase>> instances.

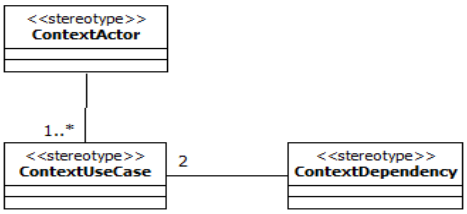

Figure 6. Representation of use case diagram constraints.

In Figure 7 we present an activity diagram constraint that shows the type of relationship that must exist between the stereotypes <<ContextActivity>> and <<ContextTransition>> and corresponding multiplicities. Indeed, each <<ContextTransition>> instance can relate exactly two <<ContextActivity>> instances.

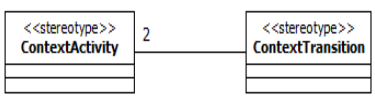

Figure 7. Representation of activity diagram constraints.

Sequence diagram constraints are shown in Figure 8. Proposed sequence diagram stereotypes are constrained by defined conditions that limit their use in the context-awareness domain. These constraints show how proposed stereotypes are associated and fix the number of corresponding instances to be used in Model level (M1). Indeed, each <<ContextObject>> instance must have only one <<ContextLifeLine>> instance and each <<ContextMessage>> instance must be related to exactly two <<ContextLifeLine>> instances because one message represent the interaction between two objects by joining their corresponding life lines.

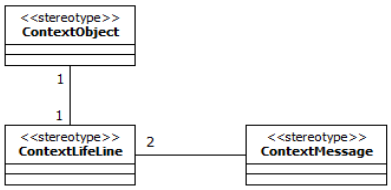

Figure 8. Representation of sequence diagram constraints.

Thirdly, as an example of expressing constraints with OCL, we take the case of two specific constraints which can be attached to all proposed stereotypes. These constraints forbid that two attributes or two operations have the same name in the same stereotype. When applied, for example, on the <<ContextTransition>> stereotype, the body of these constraints can be written with OCL language as follows:

Context ContextTransition inv:
Attributes -> forAll(Attr1,Attr2 |
Attr1<>Attr2 implies
Attr1.NameOfAttribute<> Attr2.NameOfAttribute)

Context ContextTransition inv :
operations -> forAll(Op1,Op2 | Op1<>Op2 implies
Op1.NameOfOperation<> Op2.NameOfOperation)

4.4. UML Context-aware Profile Tagged Values

UML diagrams tagged values define the properties (or attributes) of the proposed stereotypes for the corresponding diagram. Tagged values permit to differentiate the proposed stereotypes (<<ContextActor>>, <<ContextUseCase>>, <<ContextLifeLine>>, <<ContextActivity>>, <<ContextTransition>>, <<ContextObject>>, etc.,) from the existing UML metaclasses (respectively "Actor", "UseCase", "LifeLine", "Activity", "Transition", "Object", etc.,). Each tagged value is defined by a property definition (attribute name) and its possible values. Proposed tagged values will provide predefined values of specific and contextual characteristics (or properties) that cannot be modelled with standard UML notations. For example, when we apply the tagged value "State_of_user" to the stereotype <<ContextActor>>, we include predefined values such as "walking", "sitting", "standing" and

"traveling" that can be used to model many changing situations of the user's state while executing an application. In activity diagram, applying the tagged value "Context_Of_Guard" on <<ContextTransition>> stereotype means that the guard (or condition) of the proposed transition will not have the same effect as in standard UML because this tagged values will have many results according to its predefined values ("day", "night", "indoor" or "outdoor").

Table 5. Description of the proposed UML tagged values.

UML Diagram Name	Applied to (Stereotype)	Property Definition	Value Definition
Use Case Diagram	ContextActor	Actor_Type	"Human", "Software", "Hardware"
		State_Of_User	"Walking", "Sitting", "Standing", "Traveling"
		Actual_Location	"Home", "Office", "Hotel"
		Spoken_Language	"Arabic", "French", "English
		Used_Money	"Dollar", "Euro", "Dinar"
		Used_Device	"PC", "Laptop", "Mobile", "PDA"
	ContextUseCase	Usecase_Type	"Local", "Distant"
		Usecase_ sharing	"Alone", "Common"
Activity Diagram	ContextActivity	Activity_Visibility	"Public", "Private", "Protected"
		Type_Of_Sensor	"Hardware", "Software", "Both"
	ContextTransition	Context_Of_Guard	"Day", "Night", "Indoor", "Outdoor"
		Is_Periodic	"True", "False"
Sequence Diagram	ContextObject	Object_Nature	"Internal", "External"
		Object_Type	"Permanent", "Temporary"
		Object_Role	"Creator", Destructor"
	ContextMessage	Message_Quality	"High", "Medium", "Low"
		Is_Recursive	"True", "False"
		Is_Synchrone	"True", "False"
		Is_Asynchrone	"True", "False"

As an example, we suppose that we have a transition between two activities "patient" and "pharmacy" and this transition has a guard (*pharmacy.IsOpen*) that verifies if the pharmacy is open or not. Here, we see that obtained results with UML standard notations will be general and imprecise because it do not take into account that the opening time of pharmacies vary from day to night. But, this problem will find a solution by using the new notations of the proposed UML profile because it distinguishes between day and night (as tagged values) before returning the opening time results of pharmacies. In Table 5, we specify some examples of tagged values that are applied to the proposed stereotypes.

5. Implementation

As mentioned above, this work is a part of a global project that aims to construct a complete UML profile intended to context-aware application development. therefore, the implementation of concepts began when we developed the first extended notations of UML standard elements; and it was necessary to create the real file of UML profile (even for a part of diagrams) in order to be able to continue our work. For this, we note that major headlines of this implementation were exposed in our previous work [4] which focuses on extended notations of just one UML diagram (class diagram). However, we present, here, a short review of this implementation by showing an overview of how the three packages (UsecaseUML profile, SequenceUML profile and ActivityUML profiles) are created. For this, we give examples of implementing new concepts (creating profile files, creating stereotypes, creating constraints and creating tagged values) that compose these packages. Also, we note that we used StarUML software modelling platform because it is an extensible platform which supports UML language and provides excellent extensibility, customizability and flexibility [26]. In the first example, we show how the <<ContextObject>> stereotype is created by using XML (eXtended Markup Language) format as follows:

```
<STEREOTYPE>
        <NAME>ContextObject</NAME>
        <DESCRIPTION>stereotype extended from
UML metaclass 'Object' </DESCRIPTION>
        <BASECLASSES>
<BASECLASS>UMLObject</BASECLASS>
        </BASECLASSES>
</STEREOTYPE>
```

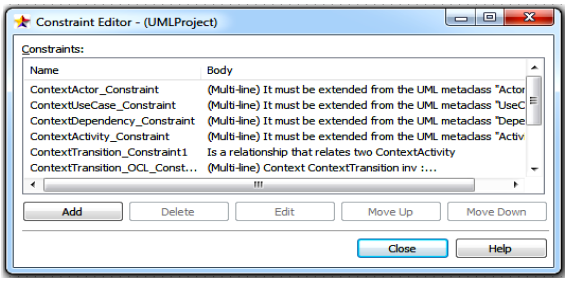

Figure 9. Adding constraints with the StarUML constraint editor.

Figure 9 shows how proposed constraints can be introduced by using the "Constraint Editor" of StarUML menu System; and we see how constraints are defined by its name and its body. Figure 10 illustrates the tagged value editor of StarUML in which some proposed tagged values are displayed with default values. These tagged values offer multiple predefined values that will be modified (or adapted) according to the current situation of a user or application.

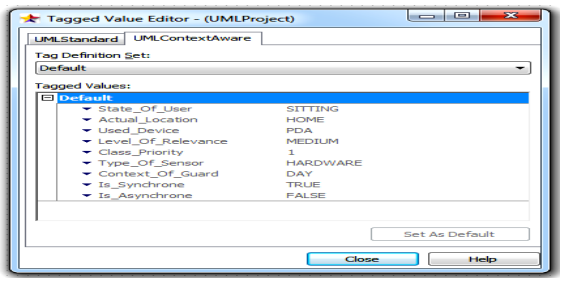

Figure 10. Specifying tagged values with the StarUML tagged value editor.

After creating all proposed extensibility mechanisms, we can extend the StarUML menu system by adding new menu items related to them. This will make easy the use of the proposed profile and will make available all new concepts to develop applications in context-awareness domain. At the end of this section, we recapitulate, in Table 7 (Appendix A), some examples of how to concretely create a profile and its extensibility mechanisms (stereotypes, constraints and tagged values).

6. Running Example

6.1. The System Description and Requirements

To demonstrate the ability of use of the proposed concepts and notations, we consider a system described by a simple example in which a same user tries to access to the same application "drug manager" in different situations (according to time, location, state of the user, nearby persons, available resources). This user can be a patient, a doctor, a nurse, a teacher, a student. Several distant services are provided by this application such as: A complete list of drugs, to buy drugs, how to consume drugs, consulting a doctor, needs a nurse etc. In this example as shown in Figure 11, we suppose that a patient is traveling from his country (A) to another country (B). During his travel, he wants to use the "drug manager" application that offers him needed information about his own drugs. From this short description, we can remark that our system is characterized by three main locations: starting location, arrival location and the way between them. When moving from a location to another one, several features are changing and a new situation is defined according to each context of use. We note that this situation may be influenced by several factors and constraints as follows [5]:

- Constraints related to the user himself (Identity, Self-Profile, Behavior, Preferences, etc.,).
- Constraints related to the application (Software, Hardware, Networks, interaction mode, etc.,) .
- Constraints related to the environment (Time, Location, Weather, Nearby persons, Surrounding objects, Available resources, etc.,).

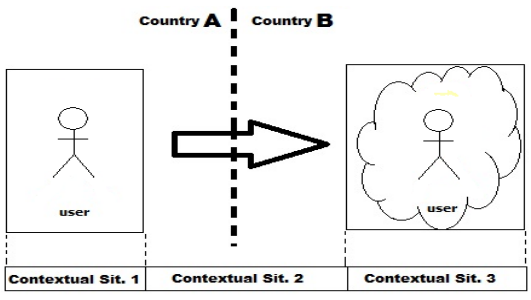

Figure 11. Example of different contextual situations.

Each of these context constraints must be represented by an adequate notation of UML language. The continuous changes identify new contextual situations, but the user must not worry about these changes because the designed application must be context-aware and must be adapted to support ubiquitous factors (anywhere and everywhere). Also, we note the take in charge of regional parameters changes (language, money, networks etc.).

Table 6. Description of contextual elements for three different situations (inspired from [4]).

Contextual Elements	Contextual Situation 01 (Country A)	Contextual Situation 02 (Border A-B)	Contextual Situation 03 (Country B)
User	Patient	Patient	Patient
State of User	Sitting, sleeping	Walking, moving	Sitting, sleeping
Location	Home, hospital	Border office	Pharmacy, hotel
Language	Arabic, French	Arabic, French English	English, French
Used Device	PC, PDA	PDA, mobile	PDA, mobile
Time	Day	Day	Night
Used Money	Euro	Euro, Dollar	Dollar
Existing Network	Internet, LAN	Internet	Internet
Nearby Persons	Doctor, nurse, parent	Friend, parent	Doctor, friend, pharmacist
Available Resources	Phone, Webcam, printer, smart TV	Phone, Fax	Fax, printer
Surrounding Objects	Medical devices	Car, bus	Table, chair

In our example as shown Table 6, we distinguish three contextual situations in which the user (as a patient) can be. Each situation is characterized by various instances (or values) of contextual elements that define the context of use related to this situation. In this case, we can consider several scenarios for our system because many elements of this system are continually changing (a long combination of contextual elements can be made to obtain scenarios). To model this example, we are going to draw UML diagrams corresponding to each of the development process steps. These diagrams will take into account all contextual information by using new concepts and notations of the proposed UML profile. After the implementation process (writing XML profile files and extending StarUML menu), our profile and its notations became available to be used.

Firstly, we lunch the profile manager of StarUML platform (Model > Profile > "profile manager") and we get the list of available profiles such as: C# Profile, C++ Profile, EJB Profile, JAVA Profile and UML Context-Aware Profile. UML Standard Profile is included as default profile as shown in Figure 12.

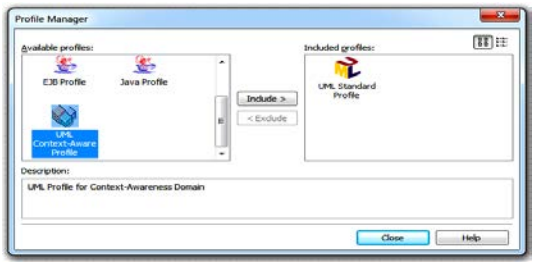

Figure 12. List of available profiles displayed by the StarUML profile manager.

Here, we choose the needed profile (in our case: UML Context-Aware Profile) and we click on the appropriate button "Include". This operation will load all related notations (stereotypes, constraints and tagged values) of the chosen profile; and at this time, this profile is completely ready to be used.

Figure 13 illustrates the "profile manager" after including UML Context-Aware Profile. Then, we choose the needed diagram from StarUML "model explorer" in order to load respective components of this diagram (including new proposed notations). Finally, we construct our diagrams by using new UML notations provided by the proposed UML context-aware profile.

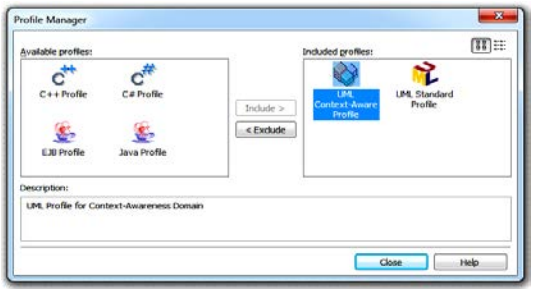

Figure 13. Including UML Context-Aware profile by using the StarUML profile manager.

6.2. Building UML Diagrams by using the UML Context-Aware Profile

Here, we expose UML diagrams that use new notations of the proposed UML profile. According to the example description, we can draw a use case diagram as shown in Figure 14 that shows how nomad and mobile actors are represented with <<ContextActor>> stereotype. This stereotype is specific to actors that are moving and that have changing features. In our example, actors such as "Patient", "Doctor" and "Pharmacist" can be represented by the stereotype <<ContextActor>> because their situation characteristics (state, time, location, device, etc.,) are continually in change. Also, we use two use cases "To consult" and "To buy drugs" stereotyped <<ContextUseCase>> because they offer several opportunities (by using attached constraints and tagged values) to represent contextual information of the use case in a clear way.

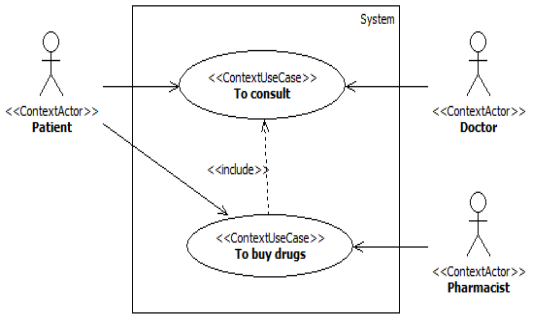

Figure 14. Use case diagram using notations of the proposed profile.

In Figure 15, we report (with short modification) a class diagram that uses extended notations cited in [4] for more understanding of the example.

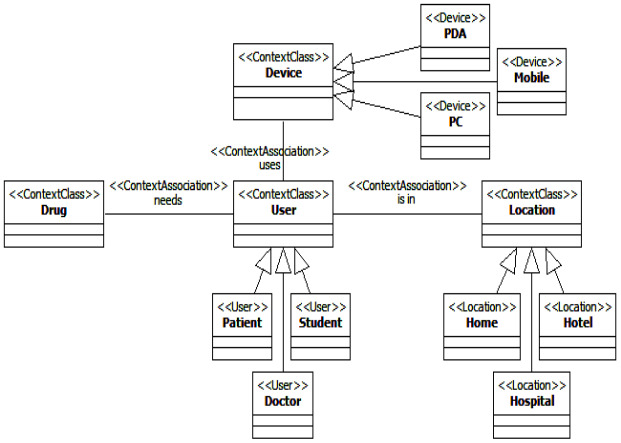

Figure 15. Class diagram using notations of the proposed profile.

Stereotypes will represent specific classes and specific associations of our example. The use of each class stereotyped <<ContextClass>> is restricted by specific conditions as "constraints" and they have appropriate attributes as "tagged values" according to the context of use. Associations between classes are stereotyped <<ContextAssociation>> to be able to relate the contextual classes stereotyped <<ContextClass>>.

In Figure 16, we present a sequence diagram of our example. The main notations of this diagram are stereotypes issued from the proposed UML profile. Objects such as "patient", "drug" and "doctor" are stereotyped <<ContextObject>> and have their appropriate life lines as <<ContextLifeLine>> stereotypes. Each of these specific objects can send or receive contextual messages considered as <<ContextMessage>> stereotypes that can be synchronous message (by applying 'Is_Synchrone' tagged value) or asynchronous message (by applying 'Is_Asynchrone' tagged value).

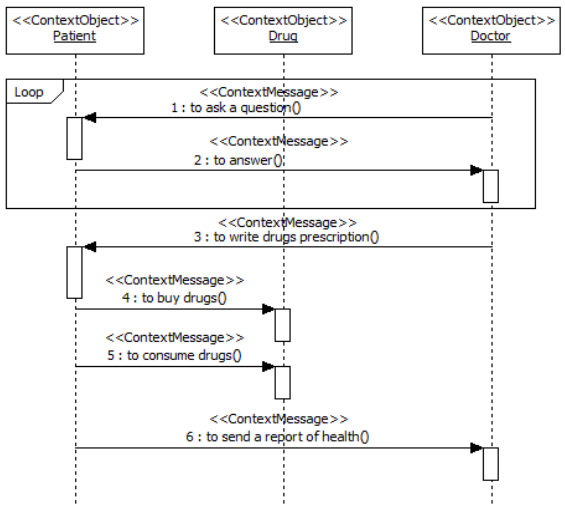

Figure 16. Sequence diagram with notations of the proposed profile.

Figure 17 presents an activity diagram in which we use new notations of the proposed UML profile to model a scenario of the example. <<ContextActivity>> stereotypes are used to model the contextual actions that correspond to operations (or methods) of a <<ContextActor>>. The progression process from an activity to another is guaranteed by a transition stereotyped <<ContextTransition>> that can be restricted by using proposed tagged values according to the context of use of this transition.

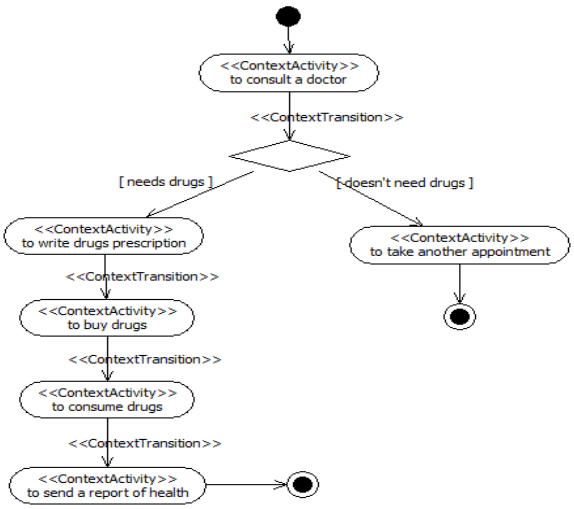

Figure 17. Activity diagram using notations of the proposed profile.

7. Conclusions and Future Works

Ubiquitous computing is an emerging field research in computing domain that assures information processing independently of time, space, device, …, etc. In Ubiquitous computing environment, we must consider changing contextual features of a system such as user's mobility, information heterogeneity and systems distribution. In this paper, we proposed an UML context-aware profile to model contextual information in ubiquitous environment. The proposed profile is destined for context-awareness domain and provides specific notations for three UML diagrams (use case, activity and sequence). This profile is a package of new notations such as stereotypes, constraints and tagged values and that are extended from existing UML elements. These new notations will complete the list of standard UML notations by modelling explicitly and appropriately all contextual situations of context-awareness domain. Proposed notations have been implemented by using StarUML software modelling platform and have been tested on a running example in the health field (drugs). As perspective, we hope implementing this profile with "Eclipse" platform which contains an extensible plug-in system to customize the environment and that provides more persistent modelling tools. Also, we hope using this UML context-aware profile to the contextual adaptation and personalization for ubiquitous information systems.

References

[1] Aldawud O., Elrad T., and Bader A., "UML Profile for Aspect-Oriented Software Development," *The 3rd International Workshop on Aspect Oriented Modelling*, Boston, pp. 1-16, 2003.

[2] Amirat A. and Oussalah M., "Towards an UML Profile for the Description of Software Architecture," *in Proceeding of International Conference on Applied Informatics*, Bordj BouAreridj, pp. 226-232, 2009.

[3] Benselim M. and Seridi-Bouchelaghem H., *Networked Digital Technologies*, Springer-Verlag, 2012.

[4] Benselim M. and Seridi-Bouchelaghem H., "Extending UML Class Diagram Notation for the Development of Context-aware Applications," *Journal of Emerging Technologies in Web* Intelligence, vol. 5, no. 1, pp. 35-44, 2013.

[5] Benselim M. and Seridi-Bouchelaghem H., "Development of Context-Aware Applications in Ubiquitous Information Systems," *in Proceeding of the 13th International Conference on Enterprise Information Systems*, Beijing, pp. 223-228, 2011.

[6] Benselim M. and Seridi-Bouchelaghem H., "Modelling Context with Extended UML," *in Proceeding of 2nd World Conference On Information Technology*, Antalya, pp. 566-571, 2012.

[7] Dey A., Abowd G., and Salber D., "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications," *Journal of Human-Computer Interaction*, vol. 16, no. 2, pp. 97-166, 2001.

[8] Djurić D., Gašević D., Devedžić V., and Damjanović V., "A UML Profile for OWL Ontologies," *in Proceeding of the Workshop on Model Driven Architecture: Foundations and Applications*, Linköping, pp. 204-219, 2004.

[9] Fuentes L., Gamez N., and Sanchez P., "Aspect-Oriented Executable UML Models for Context-Aware Pervasive Applications," *in Proceeding of 5th International Workshop on Model-based Methodologies for Pervasive and Embedded Software*, Budapest, pp. 34-43, 2008.

[10] Gherbi A. and Khendek F., "UML Profiles for Real-Time Systems and their Applications," *in Journal of Object Technology*, vol. 5, no. 4, pp. 149-169, 2006.

[11] Grassi V., Mirandola R., and Sabetta A., *«UML» 2004-The Unified Modeling Language. Modeling Languages and Applications*, Springer-Verlag, 2004.

[12] Heckel R., Lohmann M., and Th¨one S., "Towards a UML Profile for Service-Oriented

Architectures," *in Proceeding of Workshop on Model Driven Architecture: Foundations and Applications*, Netherlands, pp. 1-132, 2003.

[13] Hsu I., "An Architecture of Mobile Web 2.0 Context-aware Applications in Ubiquitous Web," *Journal of Software*, vol. 6, no. 4, pp. 705-715, 2011.

[14] Hsu I., "Extending UML to Model Web 2.0-Based Context-Aware Applications," *Journal of Software*, vol. 42, no. 10, pp. 1211-1227, 2012.

[15] Hsu I., "Visual Modelling for Web 2.0 Applications Using Model Driven Architecture Approach," *Journal of Simulation Modelling Practice and Theory*, vol. 31, pp. 63-76, 2013.

[16] Johnston S., *Rational UML Profile for Business Modeling*, Rational software, 2004.

[17] Kacem M. and Milady M., "Towards a UML Profile for the Description of Dynamic Software Architectures," *in Proceeding of Conference on Component-Oriented Enterprise Applications*, Augsburg, pp. 25-39, 2005.

[18] Kandé M. and Strohmeier A., "Towards a UML Profile for Software Architecture Descriptions," *in Proceeding of the 3rd International Conference on the Unified Modeling Language: Advancing the Standard*, York, pp. 513-527, 2000.

[19] Korherr B. and List B., *Extending the UML 2 Activity Diagram with Business Process Goals and Performance Measures and the Mapping to BPEL*, Springer-Verlag, 2006.

[20] List B. and Korherr B., *A UML 2 Profile for Business Process Modelling*, Springer-Verlag, 2005.

[21] López-Sanz M., Acuña C., Cuesta C., and Marcos E., *UML Profile for the Platform Independent Modelling of Service-Oriented Architectures*, Springer-Verlag, 2007.

[22] Luján-Mora S., Trujillo J., and Song I., "A UML Profile for Multidimensional Modelling in Data Warehouses," *Data and Knowledge Engineering journal*, vol. 59, no. 3, pp. 725-769, 2006.

[23] Object Constraint Language (OCL, OMG): OCL 2.0 Specification Version 2.0, http://www.omg.org/cgi-bin/doc?ptc/2005-06-06, Last Visited 2005.

[24] Object Management Group (OMG): MDA Guide Version 1.0.1, http://www.omg.org/docs/omg/03-06-01, Last Visited 2003.

[25] Seridi H., Bouacha I., and Benselim M., "Development of Context-Aware Web Services Using the MDA Approach," *International Journal of Web Science*, vol. 1, no. 3, pp. 224-241, 2012.

[26] StarUML 5.0 developer guide, StarUML 5.0 user guide, http://staruml.sourceforge.net/docs/user-guide(en)/toc.html, Last Visited 2014.

[27] Touzi A. and BenMessaoud M., "New Approach for Conception and Implementation of Object Oriented Expert System Using UML," *The International Arab Journal of Information Technology*, vol. 6, no. 1, pp. 99-106, 2009.

[28] Unified Modelling Language (UML, OMG): UML Infrastructure version 2.0, http://www.omg.org/cgi-bin/doc?ptc/04-10-14, Last Visited 2004

[29] Unified Modelling Language (UML, OMG): UML Superstructure version 2.0, http://www.omg.org/cgi-bin/doc?formal/05-07-04, Last Visited 2005.

[30] Van J. and Coninx K., "Using UML 2.0 and Profiles for Modelling Context Sensitive User Interfaces," *in Proceeding of the International Workshop on Model Driven Development of Advanced User Interfaces*, Jamaica, pp. 1-4, 2005.

[31] Wagner G., "A UML Profile for Agent-Oriented Modelling," *in Proceeding of the 3rd International Workshop on Agent-Oriented Software Engineering*, Bologna, pp. 1-18, 2002.

[32] Werner C., Kraatz S., and Hogrefe D., "A UML Profile for Communicating Systems," *in Proceeding of the 5th Workshop on System Analysis and Modelling*, Kaiserslautern, pp. 81-90, 2006.

[33] Ziadi T. and Jézéquel J., *Towards a UML Profile for Software Product Lines*, Springer-Verlag, 2004.

[34] Zoughbi G., Briand L., and Labiche Y., *A UML Profile Developing Airworthiness-Compliant*, Springer-Verlag, 2007.

Mohamed-Salah Benselimis PhD student atthe Department of Management Science, university of "08 Mai 45", Guelma, Algeria. He received his Master's in software engineering at Guelma University in 2009. He is a member of TWSI team at LabSTIC laboratory.His research interests include information systems,ubiquitous computing, model driven engineeringand context-awareness domain.

Hassina Seridi-Bouchelaghem received her PhD in computer science from the University of Annaba (Algeria). She is currently working as a Professor at the Computer Science Department of Annaba University and she is a teacher and a researcher at the LabGED laboratory. Her research focuses include: Information systems, Contextual Modeling, Knowledge Engineering, Semantic Web, Social Networks, and Virtual Communities.

Appendix

Table 7. Examples (extracts) of creating the proposed UML context-aware profile.

Part of Profile File	Example of Expression (Use Case Diagram)	Example of Expression (Activity Diagram)	Example of Expression (Sequence Diagram)
File Header	<?xml version="1.0" encoding="UTF-8" ?> <PROFILE version="1.0"> <HEADER> <NAME>UsecaseUMLContextAware</NAME> <DISPLAYNAME>Usecase UML Context-Aware profile </DISPLAYNAME> <DESCRIPTION>Use case UML Profile for Context-Awareness Domain</DESCRIPTION> <AUTOINCLUDE>false</AUTOINCLUDE> </HEADER>	<?xml version="1.0" encoding="UTF-8" ?> <PROFILE version="1.0"> <HEADER> <NAME>ActivityUMLContextAware</NAME> <DISPLAYNAME>Activity UML Context-Aware profile </DISPLAYNAME> <DESCRIPTION>Activity UML Profile for Context-Awareness Domain</DESCRIPTION> <AUTOINCLUDE>false</AUTOINCLUDE> </HEADER>	<?xml version="1.0" encoding="UTF-8" ?> <PROFILE version="1.0"> <HEADER> <NAME>SequenceUMLContextAware</NAME> <DISPLAYNAME>Sequence UML Context-Aware profile </DISPLAYNAME> <DESCRIPTION>Sequence UML Profile for Context-Awareness Domain</DESCRIPTION> <AUTOINCLUDE>false</AUTOINCLUDE> </HEADER>
Stereotypes	<STEREOTYPE> <NAME>ContextActor</NAME> <DESCRIPTION>stereotype extended from UML metaclass 'Actor' </DESCRIPTION> <BASECLASSES> <BASECLASS>UMLActor</BASECLASS> </BASECLASSES> </STEREOTYPE>	<STEREOTYPE> <NAME>ContextActivity</NAME> <DESCRIPTION>stereotype extended from UML metaclass 'Activity' </DESCRIPTION> <BASECLASSES> <BASECLASS>UMLActivity</BASECLASS> </BASECLASSES> </STEREOTYPE>	<STEREOTYPE> <NAME>ContextObject</NAME> <DESCRIPTION>stereotype extended from UML metaclass 'Object' </DESCRIPTION> <BASECLASSES> <BASECLASS>UMLObject</BASECLASS> </BASECLASSES> </STEREOTYPE>
Constraints	*Context ContextActor inv:* *Attributes -> forAll(Attr1,Attr2 | Attr1<>Attr2 implies Attr1.NameOfAttribute<> Attr2.NameOfAttribute)*	*Context ContextTransition inv: Transition ->forAll (Tr| self.Activity -> includesAll(Tr.Activity))*	*Context ContextMessage inv:* *Operation -> forAll(Op1,Op2 | Op1<>Op2 implies Op1.NameOfAttribute<> Op2.NameOfAttribute)*
Tagged Values	<TAGDEFINITION> <NAME>Used_Device</NAME> <TAGTYPE>Enumeration</TAGTYPE> <DEFAULTDATAVALUE>PDA </DEFAULTDATAVALUE> <LITERALS><LITERAL>PDA</LITERAL> <LITERAL>PC</LITERAL> <LITERAL>LAPTOP</LITERAL> <LITERAL>MOBILE</LITERAL> </LITERALS> </TAGDEFINITION>	<TAGDEFINITION> <NAME>Context_of_Guard</NAME> <TAGTYPE>Enumeration</TAGTYPE> <DEFAULTDATAVALUE>DAY </DEFAULTDATAVALUE> <LITERALS><LITERAL>DAY</LITERAL> <LITERAL>NIGHT</LITERAL> <LITERAL>INDOOR</LITERAL> <LITERAL>OUTDOOR</LITERAL> </LITERALS> </TAGDEFINITION>	<TAGDEFINITION> <NAME>Is_Synchrone</NAME> <TAGTYPE>Enumeration</TAGTYPE> <DEFAULTDATAVALUE>FALSE </DEFAULTDATAVALUE> <LITERALS><LITERAL>FALSE</LITERAL> <LITERAL>TRUE</LITERAL> </LITERALS> </TAGDEFINITION>