An Extension of the Use Case Diagram to Model Context-aware Applications

Ahmed Al-alshuhai
Software Technology Research Laboratory
De Montfort University
The Gateway, Leicester LE1 9BH, UK
p07143453@myemail.dmu.ac.uk

François Siewe
Software Technology Research Laboratory
De Montfort University
The Gateway, Leicester LE1 9BH, UK
fsiewe@dmu.ac.uk

***Abstract*—Context-aware applications have the ability to sense the context of the user and use the sensed context information to make adaptation decision in response to changes in the user's context. Hence, besides the functional requirements, context-awareness is an important requirement of such applications. Although, the use case diagram of the Unified Modeling Language (UML) is considered as the industrial de-facto standard for modeling the functional requirements of applications, it is insufficient to accurately capture context-awareness requirements. This paper proposes an extension of the use case diagram with new notations to cater for the modeling of context-aware applications. The proposed extension called *context-aware use case diagram* is more expressive and enables a clear separation of concerns between context-awareness requirements and functional requirements which is helpful during requirements capture and analysis of large scale or complex context-aware applications.**

Keywords—Requirement engineering; UML; use case diagram; use context diagram; context-aware applications; context information; context source

I. INTRODUCTION

Context-aware computing envisions a new generation of smart applications that have the ability to perpetually gather data about the user's context and use these data to make adaptation decision in response to changes in the user's context [1, 13, and 18]. Such applications generally run on a mobile device carried by the user and use a variety of sensors to gather context data. It will be helpful to first define the concept of context and then what it means for an application to be context-aware. Dey et al. [7] define context as "any information that can be used to characterise 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." It follows from this definition of context that the user's location, the user's activity, who the user is with, the state the physical environment (such as time, temperature, weather, and noise level), and nearby resources are important aspects of context.

Furthermore in context-aware computing, context can be classified into 3 types: (i) the user context or personal context which includes e.g. the user's id, preferences, and personal health information; (ii) the device context or ICT context which encompasses the user's mobile device capabilities, network connectivity, and battery power; (iii) and the physical environment context like time, light, location and weather. Various types of sensor are used to measure context information; these may be physical sensing devices such as a GPS (to sense user's location), a temperature sensor, and an accelerometer (to sense the user's movement); or they may be virtual sensors like the user's calendar (to sense the location or the activity of the user), and a weather web service. The generic term *Context Source* (CS) is used in this paper to refer to sensors, whether physical or virtual; while *context information* (CI) refers to the sensed data.

A context-aware application collects context information via context sources and uses this information to adapt their behaviors so as to assist the user with relevant information and services anytime, anywhere [11, 15, and 18]. Hence, the behavior of such an application is context dependent. As a result, context information and context sources play an important role in the requirement analysis of context-aware applications. In software engineering, the use case diagram of UML is commonly used to conceptualise the functional requirements of software applications. However, there is no notation in the use case diagram to represent the context; or the relationship between the context and the functions of an application. As a consequence, the usage case diagram is insufficient for accurately modeling and analysing the requirements of context-ware applications.

This paper proposes an extension of the use case diagram to cater for the modeling and analysis of the requirements of context-aware applications. The contributions of this work are fourfold:

- A new concept of *use context* is introduced to capture what CIs the behaviors of an application depend upon. Typically, a use context is a set of sequences of actions that an application must perform to acquire, aggregate, or infer CIs from raw context data produced by CSs (Sect. III).
- A novel notion of *use context diagram* is proposed to represent graphically a set of use contexts and CSs and the relationships between them (Sect. III).
- A Novel notion of *context-aware use case diagram* is presented which merges the traditional use case

diagram (that describes the functions of an application) and the new concept of use context diagram that specifies the context information the behaviors of these functions depend upon. This approach results in a rich and expressive language for the modeling and analysis of the requirement of context-aware applications (Sect. IV).

- The pragmatics of the proposed approach is evaluated using two real-world case studies (Sect. V).

II. Overview of the Use Case Diagram

There are five different diagrams in the UML for modeling the dynamic aspects of systems [17]; use case diagrams are one of them. They are central to modeling the behavior of an application and represent an excellent tool for visualizing, specifying and documenting the intended behavior of an application during requirements capture and analysis. A use case diagram is built from three basic elements: use cases, actors, and the relationships between them as shown in Fig. 1.

In UML, use cases are used to capture the functional requirements of applications. A use case describes the desired behavior of an application or part of an application (i.e. what an application or part of an application can do), without telling how that behavior is to be implemented. Use cases provide an effective way for developers to communicate with the application's end user and domain experts during the requirements elicitation and analysis phases of the software development lifecycle. Furthermore, use cases are used for validation and testing during an application development. A use case has a name and is graphically rendered as an ellipse as depicted in Fig. 1.

An actor represents a coherent set of roles that users of use cases play when interacting with these use cases [17]. Actors can be human or they can be automated systems. An actor is connected to a use case by an association (graphically rendered as a solid line) which indicates that the actor and the use case communicate with one another, possibly by exchanging messages. An actor is represented graphically as a stick figure like in Fig. 1.

There are three kinds of relationships between use cases. A *generalization* relationship between use cases means that the child use case can inherit the behavior and the meaning of the parent use case; the child may add to or override the behavior its parent; and the child may be substituted any place the parent occurs [5, 17]. The generalization relationship is represented graphically as a solid directed line with a large open arrowhead. For example in Fig. 1, "use case 2" is a generalization of "use case 3" and "use case 4". Conversely, "use case 3" and "use case 4" are specializations of "Use case 2".

An *include* relationship between use cases means that the base use case explicitly incorporates the behavior of another use case; while an *extend* relationship between use cases means the base use case implicitly incorporates the behavior of another use case. Graphically, both relationships are rendered as a dependency, stereotyped as <<include>> and <<extend>> respectively. For example in Fig. 1, the base use case "use case 5" extends "use case 1", while the base use case "use case 4" includes "use case 5".

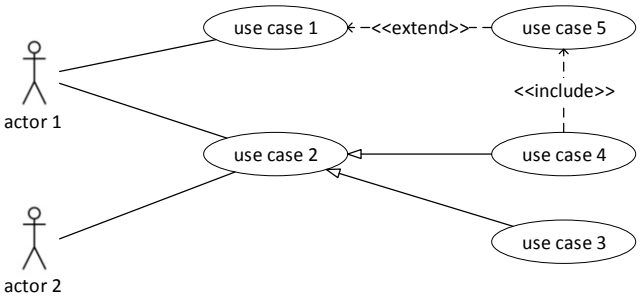

Fig. 1: an illustration of a use case diagram

III. Use Context Diagram

By analogy to use case diagrams, this paper introduces the concept of use context diagrams to model the acquisition, the aggregation and the inference of CIs relevant to a context-aware application. This helps to achieve a separation of concerns between the functional requirements and the context-awareness requirements of an application; where the context-awareness requirements specify the context information that affects the behavior of the application. A use context diagram is composed of a set of use contexts and context sources and the relationships between them, as depicted in Fig. 2.

A use context specifies the CIs that affect the behavior of an application or part of an application. It is a description of a set of sequence of actions, including variants that an application performs to acquire, to aggregate or to infer CIs from CSs. Use contexts are used to capture the relevant CIs that affect the behavior of the application under development, without having to specify how the measurement of those CIs is actually implemented. They also provide the developers a way to come to a common understanding with the application's end user and domain experts as to what CIs the application must be aware of. Similarly to use cases, use contexts serve to help validate the system's architecture and to verify the system as it evolves during development. In combination with use cases, use contexts applied to subsystems can help to design test cases for regression tests; when applied to the whole system are excellent sources of integration and system tests. A use context has a name and is graphically rendered as a dotted ellipse like in Fig. 2.

Context sources are to use contexts what actors are to use cases. Use contexts communicate with context sources to gather raw context data from which CIs are calculated. Typically, context sources are sensors; physical sensors (e.g. a temperature sensor or a light sensor) and virtual sensors (e.g. a weather web service or a calendar) alike. Graphically they are rendered as shown in Fig. 2. Context sources may be connected to use contexts only by a *context association* represented by a dashed line.

A use context may have variants. In all interesting context-aware application, there will be use contexts that are specialized version of other use contexts, use contexts that are included as parts of other use contexts, and use contexts that extend the CIs of other core use contexts. These three kinds of

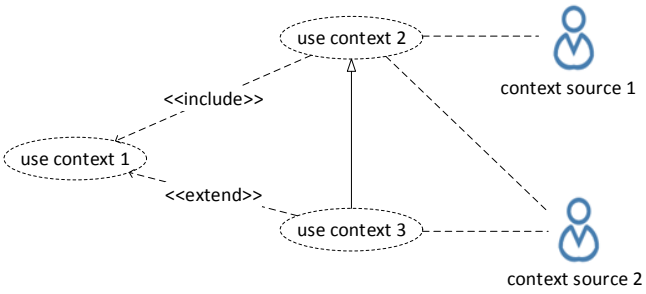

Fig. 2: An illustration of a use context diagram

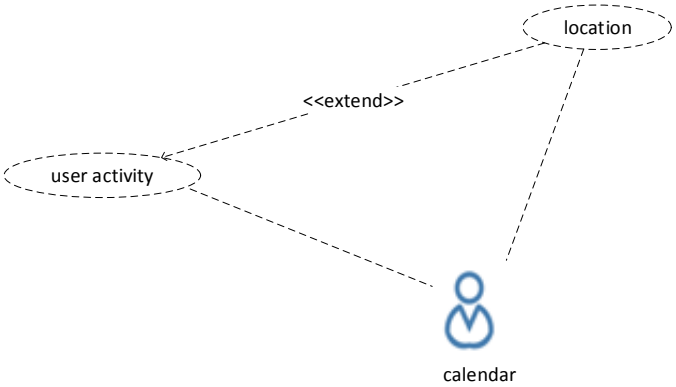

Fig. 3: Example of extend relationship

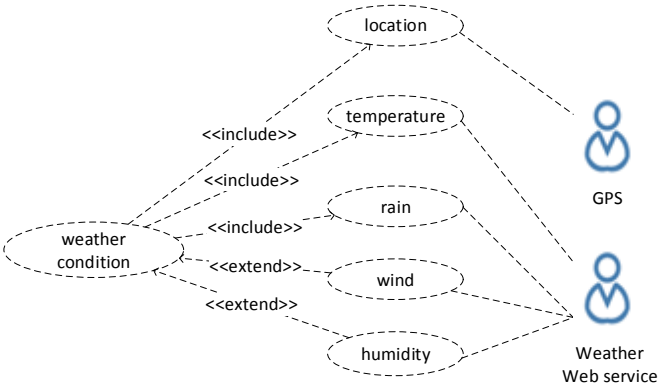

Fig. 4: Example of include relationship

relationships can be used to factor the common, reusable CIs of a set of use contexts. The same graphical notations are used for these relationships as in use cases. An include relationship is used to avoid describing the same CI several times, by putting the common CI in a use context of its own. An extend relationship is used to model the part of a use context the user may see as optional CI. In this way, optional CIs are separated from mandatory ones.

Fig. 3 shows an example of use context diagram where the use context "user activity" calculates the current user activity using information stored in the user's calendar. The extend relationship between the use contexts "user activity" and "location" means that location information might also be inferred from the user's calendar and provided as an optional CI. An example of the include relationship is given in Fig. 4 between the use context "weather condition" and the use contexts "location", "temperature" and "rain". This is to mean that the location, the temperature value and the rain status are mandatory CIs to be included in the weather reports. In the meantime, the humidity value and the wind status are not so important for the application in hand, but can be provided as optional CIs.

The following section shows how use case diagrams and use context diagrams can be combined to provide a richer and more comprehensible specification of the requirements of context-aware applications.

IV. Context-aware Use Case Diagram

A context-aware use case diagram is built from a set of use case diagrams and use context diagrams by linking use cases to use contexts using *utilize* relationships. A utilize relationship between a use case and use context means that the behaviors specified by the use case depend upon the CIs described by the use context. A utilise relationship is graphically rendered as a dependency, stereotyped as <<utilize>>, like in Fig. 5. A utilize relationship always points from a use case towards a use context. The notations used to represent a context-aware use case diagram are summarized in Table 1. Examples of context-aware use case diagrams are detailed in the following section.

V. Case Studies

This section presents two examples to illustrate how the proposed approach can be used in practice. The first example constructs a context-aware use case diagram for a smart weather application; and the second one presents a context-aware use case diagram for a navigation application.

Table 1: Notations for describing a context-aware use case diagram.

Context-aware use case diagram notations	
Context related notations	UML notations
 Context source	 Actor
 Use context	 Use case
Context association	Association
Include relationship	Include relationship
Extend relationship	Extend relationship
Generalization relationship	Generalization relationship
Utilize relationship	

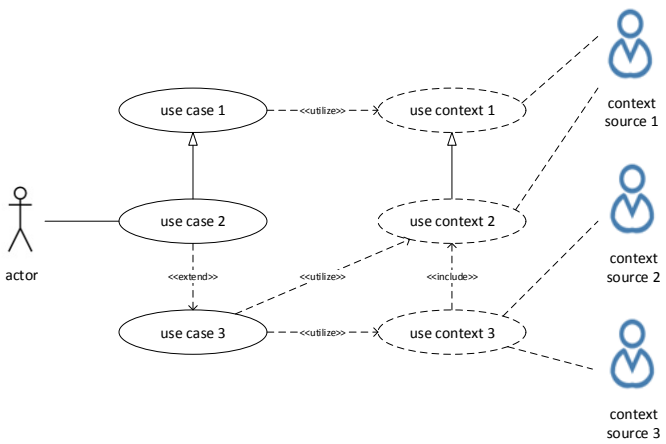

Fig. 5: An illustration of a context-aware use case diagram

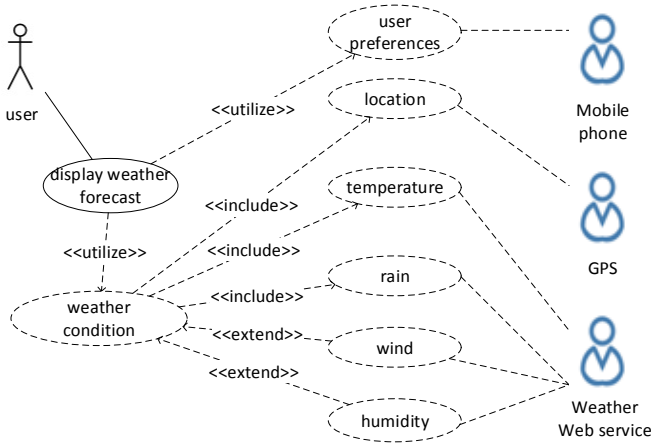

Fig. 6: A smart weather forecast application

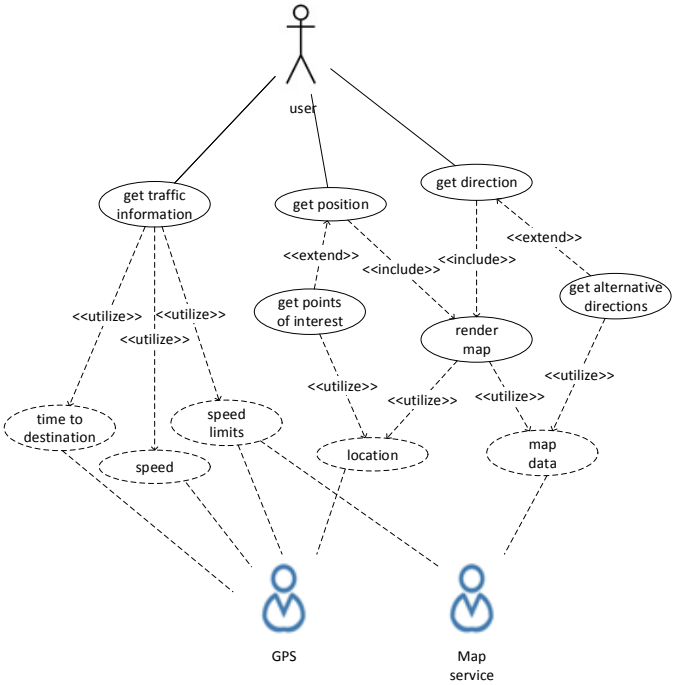

Fig. 7: A Navigation Application

A. A Smart Weather Forecast application

Consider a smart weather forecast application that runs on the user mobile phone and is aware of the user's preferences and location. The user's preferences are stored in the user profile on the mobile phone and the location information is gathered from a GPS module embedded in the mobile phone. It is assumed that the actual weather data are provided by a weather wed service, given the limited computational power of a mobile phone. Fig. 6 shows the context-aware use case diagram for displaying the weather information on the user's mobile phone based on the user's location and preferences.

The use contexts specify what CIs are calculated using the data provided by the corresponding CSs. For example, the use context "user preferences" indicates what information stored in the user profile is relevant for this application. The use case "display weather forecast" utilizes the use context "user preferences" and the use context "weather condition" to display the weather information according to the user's preferences (e.g. font size, colors, etc.). An include relationship is used to indicate that the user's location, the current temperature value and the rain status are mandatory CIs for this application. However, the wind status and the humidity data are rather optional CIs; hence the extend relationship is used for the corresponding use contexts.

B. A Navigation Application

The main functionalities of a navigation application [9, 12, 14] include give directions; suggest alternative directions in the case an accident has occurred or the road is blocked; provide traffic information such as speed, speed limits, and approximate time to destination; and show the current position of the user and the nearby points of interest; to name but a few. The context-aware use case diagram in Fig. 7 shows the use cases corresponding to these functionalities and the use contexts representing the CIs these functionalities depend upon and the CSs that will provide the raw data for computing these CIs. Hence, the use case "get traffic information" calculates the speed and the time to destination using information from a GPS module; and determines the speed limits using the location data from the GPS and the road information provided by a map service. It is assumed that the map service (e.g. Google map) provides all the data necessary to render the road network, including location coordinates of speed cameras, types of road and speed limits.

The use case "get position" displays the position of the user on the map and optionally the points of interest in the vicinity. Therefore, the use case "get points of interest" extends the use case "get position". In order to render the user position and possibly the nearby points of interest, the use case "get position" invokes the use case "render map" which utilizes the location information from a GPS and the map data provided by a map service to do so. Likewise, the use case "render map" is also invoked by the use case "get direction" which shows the route to destination and optionally suggests alternative directions.

VI. Related Work

UML is a diagram language which enables designers of information systems to illustrate high level system requirements, using use case diagrams, and to demonstrate low level system requirements, using activity diagrams [6, 8, and 10]. Researchers have recently developed new UML diagrams in their attempts to make the development of context-aware applications easier. Choi and Lee [3] proposed a model-driven approach that uses UML's use case diagrams to elicit the requirement of context-aware applications. In particular, the approach helps analysts and stakeholders pay more attentions to context related issues such as system platform, target users, intelligence, possible context-aware services and agreement with other stakeholders, and understanding contexts with decision tables and trees.

ContUML [2] is a UML-based language for model-driven development of context-aware applications. However, ContUML essentially extends the UML's class diagram with special classes for CIs and context-awareness mechanisms. Our context-aware use case diagrams are more abstract than class diagrams and so more suitable for requirement elicitation and analysis. It is understood that ContUML may be used for the realization of context-aware use case diagrams during system development. Almutairi et al. [4, 16] extended the UML's use case diagram and class diagram to capture the security requirement of context-aware application. In particular, they introduces a "requires" relationship between a use case and CIs to indicate the CIs the behaviors described by that use case depend upon. In our approach, use context diagrams are used to specify CIs and their corresponding CSs; separately from the use cases that will utilize those CIs. This separation of concerns between functional requirements and context-awareness requirements is helpful, especially when dealing with large scale or complex context-aware applications.

VII. Conclusion

This paper presented a new extension of the use case diagram of UML to cater for both the functional requirements and context-awareness requirements of context-aware applications. This novel extension is called a *context-aware use case diagram*. Indeed, the concept of *use context* is introduced to specify the CIs that the behaviors of the application under development depend upon. Then a notion of *use context diagram* is proposed to depict graphically the relationships between a set of use contexts and the CSs. While use cases capture the functional requirements, use contexts describe the context-awareness requirements. Such a separation of concerns is helpful during system development. A context-aware use case diagram is built from a set of use case diagrams and use context diagrams by linking use cases to use contexts using the *utilize* relationship. A utilize relationship between a use case and a use context means that the behaviors specified by the use case depend upon the CIs described by the use context. The pragmatics and flexibility of the proposed approach are illustrated using two case studies.

In future work, other UML diagrams, such as the activity diagram and the class diagram, will be extended in an effort to ease the development of context-aware applications.

References

[1] Y. Chen and C. Petrie. "Ubiquitous Mobile Computing". IEEE Internet Computing, 7(2):16–17, 2003.

[2] Q. Z. Sheng and B. Benatallah. "ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services". In Proceedings of the International Conference on Mobile Business (ICMB'05), Sydney, Australia, 2005.

[3] J. Choi and Y. Lee. "Use-Case Driven Requirements Analysis for Context-Aware Systems". In: the Future Generation Information Technology Conference, Springer, Heidelberg, 2012.

[4] S. Almutairi, A. Abu-Samaha, G. Bella and F. Chen. "An enhanced Use Case diagram to model Context Aware System". Science and Information Conference (SAI), London, UK, 7-9 October 2013.

[5] K. Henricksen and J. Indulska. "A Software Engineering Framework for Context-Aware Pervasive Computing". In Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications (PerCom'04), Florida, USA, 2004.

[6] D. Ayed and Y. Berbers. "UML profile for the design of a platform-independent context-awareness applications". Proceedings of the 1st workshop on MOdel Driven Development for Middleware (MODDM '06), Melbourne, Australia, 2006.

[7] A. K. Dey and G. D. Abowd. "Towards a better understanding of context and context-awareness". Lecture Notes in Computer Science; Vol. 1707, 1999.

[8] A. Finkelstein and A. Savigni. "A Framework for Requirements Engineering for Context-Awareness Services". First International Workshop from Software Requirements to Architectures (STRAW 01), 2001.

[9] O. Masreiter and E. Metzker, "A context-driven use case creation process for specifying automotive driver assistance systems", IEEE International Requirements Engineering Conf., pp. 334-339, 2004.

[10] D. Skogan, R. Gronmo, and I. Solheim. "Web Service Composition in UML". Proceedings of the 8th International IEEE Enterprise Distributed Object Computing Conference (EDOC'04), California, USA, September 2004.

[11] D. Riboni and C. Bettini. "OWL 2 modeling and reasoning with complex human activities". Pervasive and Mobile Computing, 2011.

[12] S. Saeedi, N. El-Sheimy, X. Zhao, and Z. Sayed. "Context-Aware Personal Navigation Services using Multi-Level Sensor Fusion". In Proceedings of the 24th International Technical Meeting of the Satellite Division of the Institute of Navigation, USA, September 2011.

[13] K. M. Feigh, M. C. Dorneich and C. C. Hayes. "Toward a characterization of adaptive systems a framework for researchers and system designers". The Journal of the Human Factors and Ergonomics Society, 2012.

[14] M. Madkour, D. E. L. Ghanami, A. Maach et al. "Context-aware service adaptation: an approach based on fuzzy sets and service composition". Journal of Information Science and Engineering, 2013.

[15] J. Choi. "Context-driven requirement analysis". In Computational science and its application. Springer, Heidelberg, 2007.

[16] S. Almutairi, A. Abu-Samaha and G. Bella. "Specifying Security Requirement for Context Aware System using UML". In Seventh International Conference on Digital Information Management (ICDIM) 978-1-4673-2430-4/12/ in Macau, 2012.

[17] G. Booch, J. Rumbaugh and I. Jacobson. The Unified Modeling Language User Guide. Addison Wesley, 1999.

[18] M. Weiser. "The computer for the 21st century". SIGMOBILE Mob. Comput. Commun. Rev. 3(3), 1999, pp. 3-11.