An Extension of Class Diagram to Model the Structure of Context-Aware Systems

Ahmed Al-alshuhai * and François Siewe **
* Software Technology Research Laboratory, De Montfort University, Leicester, United Kingdom
p07143453@myemail.dmu.ac.uk
** Software Technology Research Laboratory, De Montfort University, Leicester, United Kingdom
fsiewe@dmu.ac.uk

Abstract: Context-aware systems (CASs) have become a reality thanks to the development of smart software and hardware to assist the users in various real life activities. The proliferation of context-aware services has led to the emergence of environments where services are made available for usage anywhere and at any time. CASs have the ability to capture users' contexts and use their instance values to provide self-adaptive services in response to context changes. Modelling and documenting the structure of such a system during the design phase is vital for system validation, testing, maintenance and version management. The Unified Modelling Language (UML) is the de facto industrial standard for system modelling and development. The UML class diagrams provide notations for modelling graphically the structure of a system in terms of classes and the relationships between them. However, these notations are insufficient to model the structure of CASs. This paper proposes a new set of notations to represent context and context-awareness and their relationships with classes in class diagrams. Hence, the structure of CASs can be specified, visualized, constructed, and documented distinctively during system development. The proposed approach is evaluated using real-world case studies.

Keywords: Software Requirement Engineering (SRE), Unified Modelling Language (UML), Context Aware System (CAS), class diagram, context class diagram, context-aware class diagram.

Introduction

Typically, the design stage of software systems usually uses a class diagram to specify the system structure components and their relationships, which illustrates the system functions in preparation for the coding stage and completing the system implementation [13 and 16]. In the meantime, the class diagram is important in the design stage, which provides information and attributes' connections and outlines clear models of the main objects for the system, and also specifies the relationship types between objects (classes), helping to decrease the complexity of the system structure. The system designers and developers should focus carefully on the interaction between UML diagrams' connections of attributes, classes, objects and methods and their relationships. It is also important that UML diagrams be integrated with each other to create comprehensive maps of any system in preparation for the construction and implementation stages without any errors [6, 7 and 17].

Context-aware systems (CASs) have become a reality thanks to the development of smart software and hardware to assist the users in various real life activities. CASs have the ability to capture users' contexts and use their instance values to provide self-adaptive services in response to context changes. Modelling and documenting the structure of such a system during the design phase is vital for system validation, testing, maintenance and version management. The UML class diagrams apply the concept of the object-oriented model to express the components' structure for CASs, but this structure requires extension to represent explicitly context awareness requirements as entities and their properties and functions [18]. This paper proposes a new approach that extends UML class diagrams with graphical elements to express distinctively the context-awareness aspects of CAS's structure. Namely, (i) the notion of context is clearly represented as a special kind of class to store and manipulate context information (CI); and (ii) context-awareness is modelled by a special relationship between a class and the context that class depends upon. The contributions of this work are threefold:

- A new concept called context class is introduced to represent the structure of *context objects*. Context objects store CIs sensed by context sources (CSs) or aggregated from sensed data. A *context class diagram* is set of context classes and the relationships between them (Sect. III). These relationships are the same as in traditional UML class diagrams.
- A novel notion of *context-aware class diagram* is presented which extends the traditional UML class diagrams to enable the representation of context information that affect the behaviors of a class. A new relationship, called *utilization*, between a UML class and a context class is used to model context-awareness; meaning that the behaviors of the UML class depend upon the CI represented by the context class. Hence a context-aware class diagram is a

rich and expressive language that distinctively depicts both the structure of classes and that of the CIs they depend upon (Sect. IV).

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

Related Work

Traditional UML Class Diagram

A UML class diagram is a set of classes and the relationships between them. It is the most common diagram used in object-oriented modelling for visualizing, specifying and documenting structural models and schemas. A class has a name, attributes and methods as depicted in Fig. 1. Access to attributes can be restricted using visibility markers. There are five visibility markers: a private attribute is visible only inside the class and so hides its information from other classes; a public attribute is visible to other classes; a protected attribute is visible in sub-classes; a derived attribute is used to compute values from different attributes; a package attribute is visible to all classes within the same package [3, 11 and 15]. There are three main types of class relationships, and each one has a special meaning as follows: (i) dependency relationships represent using relationships among classes; (ii) generalization relationships model inheritance between a parent class and a child class; (iii) association relationships represent structural relationships among classes. There are three types of association relationships. The most common association relationship type is multiplicity which specifies how many objects are used between different classes, such as one-to-one relationships and one-to-many relationships. The second association relationship type is aggregation, which refers to the relationship between a whole and a part. The third type of association relationship is called composition, which is a special kind of aggregation which links the lifetime of the whole and its parts [3].

CAS Structure Modelling

System design, modelling and simulation are important phases before the actual implementation of any system. However, Simulation of CAS will define which part is important for CAS, specify the purpose of any interaction with an external part and know different options for CAS responses. Furthermore, system structure simulation demonstrates the whole architecture of the system and helps to depict the physical system as models and the relationship between them. CAS structure simulation is necessary to express the different parts connected with CAS and to depict parts in relation to each other, representing a set of objects or components.

Existing Approaches using UML Diagrams

Researchers have recently developed new UML diagrams in their attempts to make the development of context-aware systems easier.

C. Simons [2] proposed UML extension called the Context Modeling Profile for context models in mobile distributed systems. The resulting models visualize Meta-information of the context (such as source, validity). The author suggested a case study of meeting system include stereotypes to illustrate system consists and represent context-association with Meta-information for each context that help modeler to provide values such as the association between person and activity.

Sheng and Benatallah [5] used the Context-UML approach of Class diagram to describe the development of the initial stage of a complete model-driven for Context Awareness Services: they modelled CASs by Context-UML and their implementations were automatically generated and then deployed. The Context-UML Approach has tried to resolve the lack of formalizing of the development of CASs. The Context-UML Approach uses stereotypes to simplify and represent system requirements. The authors do not investigate CS and disguise the complexity of context acquisition from CASs designers, and focus on the functionalities of CASs, rather than context sensing.

Alghathbar et al [9] published a paper on securing UML information flow using FlowUML. The authors investigated a new approach of FlowUML to specify the information system specifications to be validated based on UML.

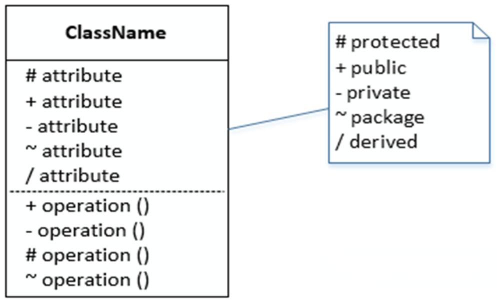


Figure 1. Illustration of a UML class

Masreiter and Metzker [8] tried to depict CAS functionalities and behaviour as well as CI situations using existing use case diagram notations; they attempted new processes for CI situations, aiming to capture and specify CAS requirements, but they did not identify real situations between user needs and CAS functionalities to express the CI, CS and CAS interactions.
Choi and Lee [19] used the model-driven approach, as depicted in the example in Figure 2.9, which explained the extension of the Use Case diagram to model CASs by UML. They specified CI such as temperature ranges to produce actions, although this research did not express them in the use case diagram or explain how CASs are affected by CIs and CSs. Additionally, the CAS functionalities and scope were not sufficiently investigated, as this is one of many studies using UML to model a set of functionalities and the scope of a CAS.
Almutairi et al. [20] enhanced the use case diagram to model CAS. They tried to adjust the new shapes used (dotted shapes for actors and use cases) if the use case required other use cases as CI to extend the main use case and to specify CI factors such as time and location, although the extended shapes were still limited to expressing the CAS interactions between CS and CAS. On the other hand, there is a lack of modelling as a consequence of CAS; the enhancement of the CAS involves complex activities such as research into UML, which needs to be extended by specifying certain CAS structures and behaviours.

Context Class Diagram

This section provides a new concept of context class diagram which suggests a new shape called context to design the context's properties and operations, using the new shape to express the contexts and their values. The new concept of the context class diagram provides an effective way to describe the context structure to complete the structure design of CASs. The basic class diagram represents context objects as classes and links them via the same relationships but contexts are still not represented. Also context entities cannot be stored in CASs databases, which always changing. Furthermore, context values are always changing, which affects the system results. In addition, the difference between the design stage of normal system and CAS is that the parameters of the system may be stored in databases or inserted by the user through system run-time, but in CASs the situation is not traditional: it needs to retrieve CI from the CS by itself and the contexts change in response to the user and cannot to be stored in CASs' databases (the last output of CASs may be stored in the database without awareness CI changes). To support this concept of the context class diagram representing the context properties and functions, a new shape is used (as depicted in Fig. 2), graphically represented as a hollow diamond dotted shape divided into three parts.
Context shape is divided into three partitions as follows:

- Context name: the top partition is specified for context name, which is written in capital letters in the center. A name is specified for each context to simplify the design stage of CASs' structure, which outlines entities affecting CASs' services.
- Context attributes: the second partition in the middle of the context shape is for attributes to list all context properties which affect classes' properties. Specifying context properties in the design stage also helps to simplify the component structure of CASs. Examples of context attributes include int, float, bool, string and so on.
- Context functions: the third partition is at the bottom of context shape to list the functions calculating the values of CI to fulfil the needs of CASs' services, such as retrieving the value of traffic status or the value of remaining time.

A Context-Aware Class Diagram

The context aware class diagram approach extends the class diagram and uses two concept elements which merge the existing elements of the class diagram and the new elements of the context class diagram. It also suggests a new relationship, called utilization. The proposed new concept of the context class diagram with a new shape is important to support the merging of the existing elements of the class diagram to represent clear structural modelling of CASs. Another important aspect of the context class diagram is that it uses the special relationship of utilization, which allows CASs' classes to interact with contexts to retrieve the values of CI.

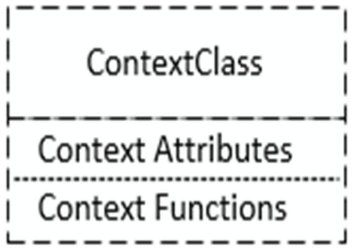
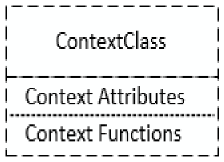
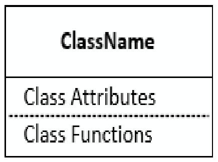


Figure 2. An illustration of a context class

Table 1. Context-aware class diagram elements

Context related elements	UML elements
 Context-class	 Class
——▷ Generalization	——▷ Generalization
—— Association	—— Association
——◇ Aggregation	——◇ Aggregation
——◆ Composition	——◆ Composition
- - - - - - - - > Dependency	- - - - - - - - > Dependency
- - - - - - - - - ▷ Utilization	

Continuous retrieval is the solution to capture the context values in the user environment, which are constantly changing. Furthermore, this approach outlines the difference between class functions and context functions: class functions are executed by CASs, while context functions are executed by CSs. The elements used to represent a context-aware class diagram are summarized in Table 1.

In Fig. 3, we can observe the both shapes of class and context are merged for the context-aware class diagram approach and linked using a new relationship –the utilization approach– which fulfils the context objects by retrieving the values of CI. In a simple definition of the context-aware class diagram approach, the concept of the class diagram creates a summary of all context objects for CASs and the concept of the context class diagram creates a summary of all context entities for CASs.

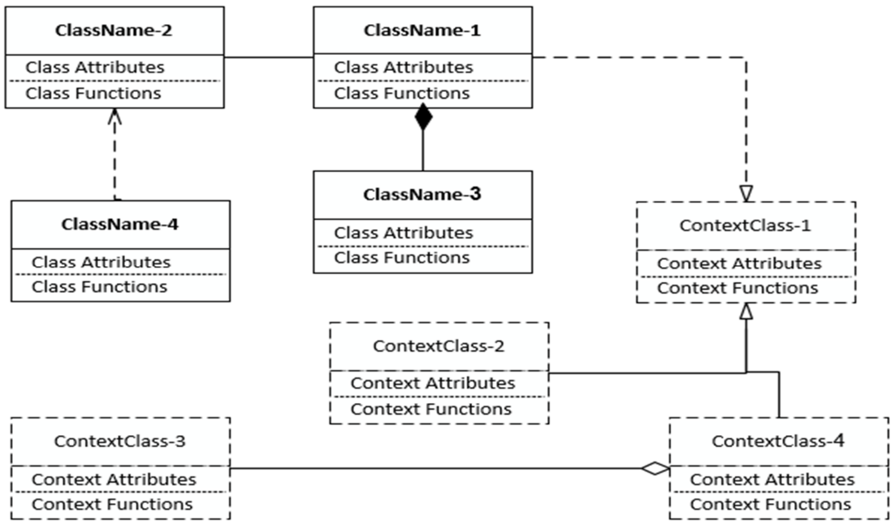


Figure 3. Illustration of a context-aware class diagram

The context-aware class diagram approach can answer the following questions:

- What must the CS do for context objects?
- How will the CAS fulfil the context awareness requirements?

- What classes will need to implement CASs objects that meet context awareness requirements?
- What properties and operations will each class have?
- What proprieties and operations will each context have?
- How will the classes and contexts interact with each other?

Specifying the Class Structure for CASs' Components

The basic classes have the ability to specify the objects for structure components to design CASs. Traditional systems are well designed by class diagrams to express the system functions to store different services and the necessary parameters for normal systems in the databases for multiple uses; conversely situation in CASs cannot be completed without context values, which are collected by different CSs to represent the context awareness requirements to affect the CASs outputs. However, using existing elements of the class diagram, it is possible to model the CASs components and their functions and properties by adjusting the usage of class diagrams to model the static structure of CASs. In addition, designing the context objects as classes helps to apply CASs functions, and each single function is depicted by a class shape and may retrieve other classes' data or apply inheritance interactions for other classes using set types of relationships to achieve the calculated values to fulfill the context awareness requirements of CASs. Furthermore, in the context-aware class diagram, the existing classes' usage is adjusted to design the user preferences and set their properties, and the context objects can be structured by class diagrams. In other words, the component structure of CASs can be designed using the elements of the class diagram to design the user and system components as classes.

To support this concept of the adjustable class diagram, the diagram shown in Fig. 4 above expresses the practical usage of existing elements of the class diagram to represent the class partitions of properties and operations. An example of a render engine is depicted and this is one class of navigation system classes. A render engine is a class visualized by the existing class shape as a context object rather than a context entity.

Specifying the Context Structure for CASs Components

The context-aware class diagram approach suggests a new shape to represent the context aspects, such as context attributes and operations. However, the main reason to specify the context's parts and structure is to design the real structure of context components, which are sensed by different CSs. Classes and contexts interact with each other for certain purposes: CASs' classes are designed to retrieve the CIs values from CSs. This approach provides solutions for CAS designers, who should focus on the availability parameters produced by CSs to fulfill CASs' requirements of context awareness and user needs. In addition, context shapes are designed to outline the exact CIs values needed for CASs' structure design. CS creates the physical entities as contexts to make the CASs' parameters, which might be atomic CI or composite CI.

This approach also investigates CS properties and functions that produce set types of CI values. In other words, the context-aware class diagram approach expresses the context values in the design stage of CASs to identify the parameters that should be retrieved for classes of CASs. The shapes of both classes and contexts will decrease the complexity of the structure of CASs as simple models in the hierarchy elements of classes and contexts with the same relationship types used in the class diagram, with a new relationship of utilization to link the classes (context objects) with contexts (context entities).

In Fig. 5, we can observe the both shapes of class and context are merged for the CACD approach and linked using a new relationship – the utilization approach – which fulfills the context awareness objects by retrieving the values of CI.

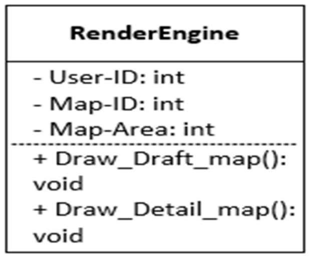


Figure 4. An example of the adjustable elements of class diagram

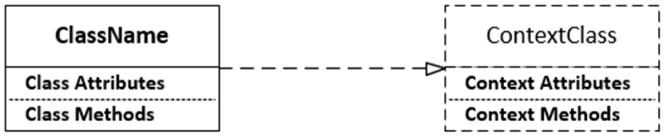


Figure 5. An illustration of elements for the context aware class diagram

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 class diagram for a weather forecast system; and the second one presents a context-aware class diagram for a navigation system.

A Weather Forecast System

To support the extension approach of context-aware class diagram, this section provides an example of a Weather Forecast System (WFS) using the elements of both the adjustable class diagram and the context class diagram. This example of this approach sets out to design the main components of the WFS to express the effectiveness of new elements of this approach. In addition, the WFS provides a set of services to create awareness of many types of weather conditions, such as rain status, temperature, snow status and so on. This kind of system can be configured on mobile devices, which have the ability to support user needs at any place and any time. The location of the user is the most important CI and the key to which other CIs need to be known: for example the services of WFS provide information on temperature, snow status, humidity, cloud status, rain status etc. for a specific location, which means that the location of the user is the target for which it is necessary to be context aware for different types of information. Furthermore, the user’s mobile phone can store the user’s preferences for future uses depending on the location of the user, and GPS can be used to gather the user’s location. This example aims to explain how, when designing a WFS, specific CIs are needed for the CAS design stage through capturing functions set by different CSs. Typically, the structure of a WFS consists of three components merged into one system that appears in the user profile to give powerful performance to the mobility services that hide the backbone of intelligent systems such as WFS.
The above diagram (Fig. 6) used basic elements of class diagram and their relationships to design the structure components of WFS, and also the class shape to outline all objects of WFS and their properties and operations.
This example investigates (as designed in Fig. 7) the structural components of the WFS as follows:

User profile
In smart systems, the user’s preferences and location control the services and final actions for users. WFS users can find out about the weather conditions as real life weather. However, this component of the user profile can be structured using the basic classes to identify the user properties and to classify the user’s account and services. In addition, the user information is stored in the mobile phone as preferences such as user location, which are gathered from CSs and embedded in the mobile phone.

WFS domain
The component of the WFS outlines the main context objects of weather which specify the services needed for users, such as the search engine for the location. However, the objects of the WFS set the structure for retrieving information automatically for users depending on the availability of CSs and their CIs.

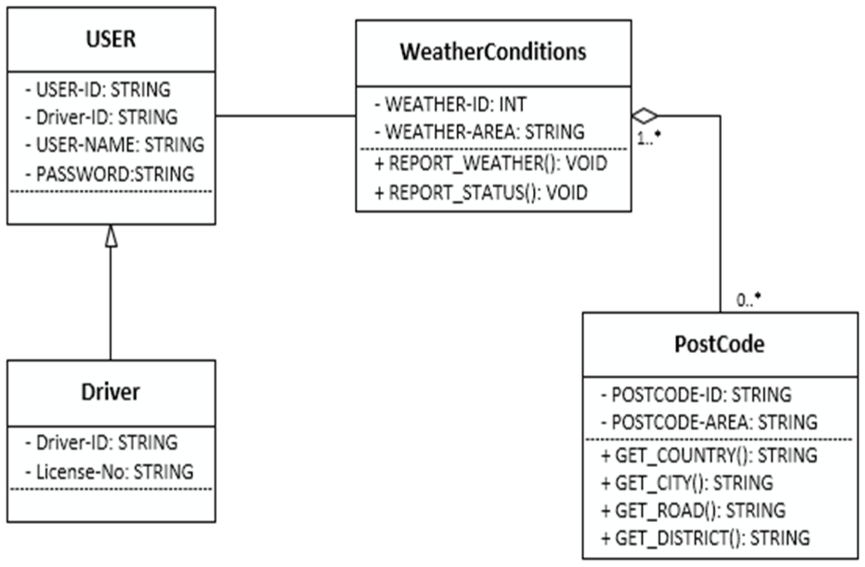


Figure 6. An illustration of classes for the WFS

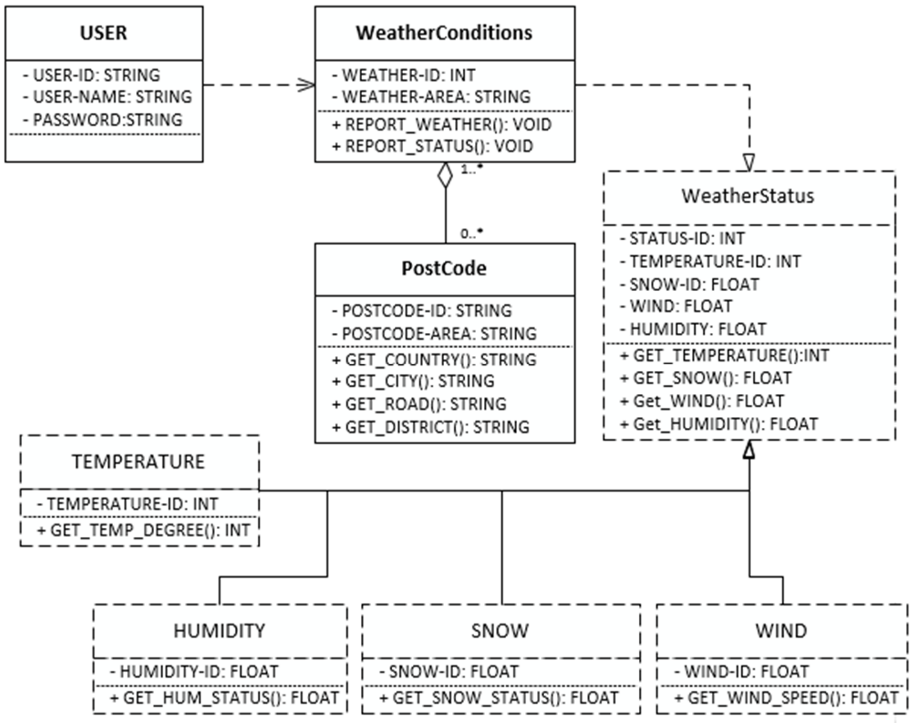


Fig. 7: The structure components of a WFS

In addition, the component structure of the WFS specifies which parameter should input to the system (depending on user preferences); then the WFS will interact with suitable CS to retrieve the requested information as system parameters for specific functions to produce an action for users.

Context service

The context service (context source) collects the parameters (context information) for the WFS. These parameters may be atomic CI or composite CI. In addition, CS provides automatic parameters for WFS to produce the CIs values required to fulfill the context objects (WFS operations should receive specific CIs values to be executed). The context-aware class diagram approach provides a new shape for contexts as single CI (atomic CI) or multiple CI (composite CI) to complete the operations of the context objects. Furthermore, different types of CSs are suitable to produce CIs for WFS, such as GPS, Global Weather, Weather Web Service and so on.

A Navigation System

Services of Navigation System (NS) are currently running a series of global services to produce virtual platform guide for users [14]. The main CS is GPS, used to provide NS for monitoring a variety of systems and updating changes of CIs. Additionally, GPS is used for sensing and monitoring important objects such as location, direction, speed and so on.

GPS reading and sending messages through the GPS network are required to monitor the movement from node to node to follow and update these changes. Several CSs aim to evaluate monitoring procedures and follow up the CIs' changes.

NS is the system required when a journey is taken in the least possible time using the most accurate and current method of reaching the destination. In order for the NS to be accurate, the following information is needed: start point; end point; direction of journey using the most convenient route; and distance of journey with remaining time, factoring in alternative options such as suggested routes in case of road traffic accidents or other unexpected events. The classes and contexts in Fig. 9 show the main components of NS. There are two main context objects: one for acquiring duration updates and one for rendering a map or retrieving the appropriate map for a specific journey. These need many single CIs: speed, direction, position coordinates and map element details provided by GPS or web services (such as traffic services). Fig. 8 shows how the context-aware class diagram approach demonstrates the main facilities of NS updates themselves, depending on CI, and usually changes depending on the user environment. NS is a smart system informing users by GPS; GPS is a context service, not a context-aware system used within a location based on a CAS acting as a sensor.

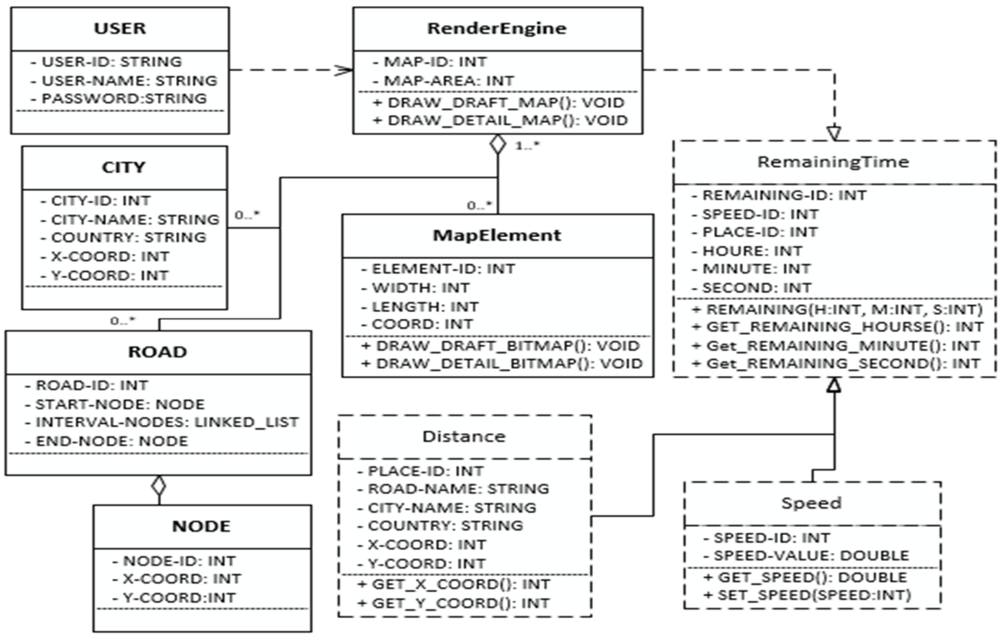


Figure 8. The structure components of a Navigation System

A Traffic System

Today, transportation traffic is a real challenge for many countries and governments are spending huge budgets on extending roads and other services to resolve road traffic problems. Some countries are constructing smart roads which record traffic movements using sensors and specify any accidents. Road movement is the most important information that should be extracted and carried to the traffic system to alert drivers to road situations. The traffic system provides traffic information such as speed, speed limits, and other services about road situations and movements. The classes and contexts in Fig. 9 are designed using the context aware class diagram approach to provide different traffic services such as road movement simulation of individual cars. It is a complex task to capture all travelling cars' position and behavior within roads and express accurate information, which usually provides a huge amount of data and requires an accurate simulation to depict cars, roads, junctions and networks and their situation and behavior. The above diagram illustrates an example of class diagram using a basic shape to design the user and driver as context objects. The relationship between the driver and the user is generalized to identify the inheritance relationship between parent and child classes.

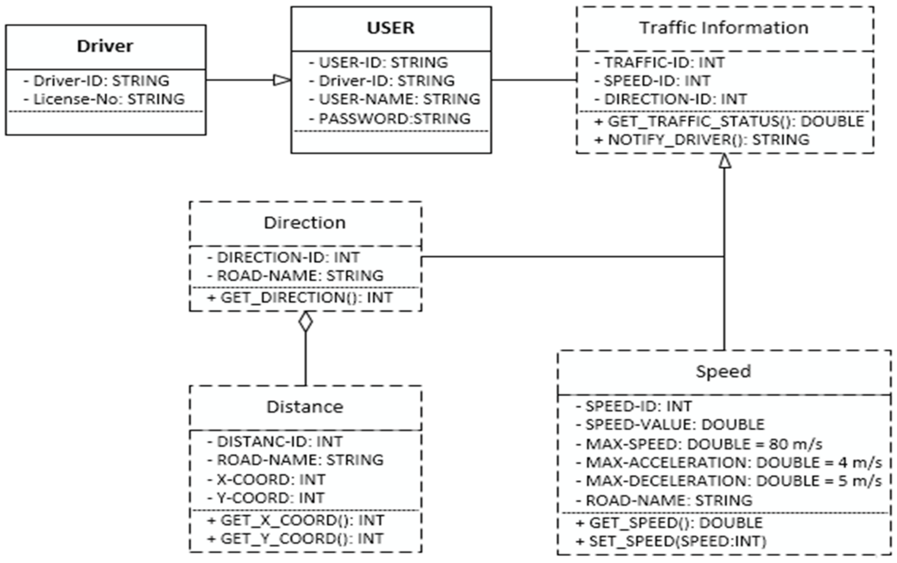


Figure 9. The structure components of a Traffic System

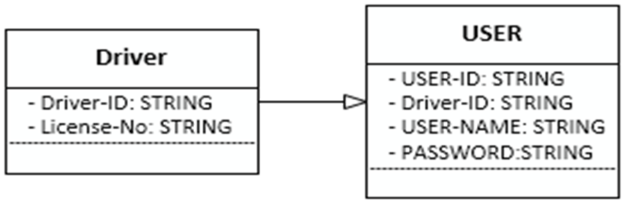


Figure 10. An illustration of classes for the traffic system

Conclusion

This paper presented a new extension of the class diagram of UML to cater the structure components of CAS. However this approach has extended the class diagram to design the structure of CASs; which includes two concepts of adjustable class diagram and context class diagram. Both concepts' elements are merged for the context-aware class diagram approach to design all components of the structure of CASs. In addition, this paper has investigated the usage of each concept and when their elements are used; these concepts connect their elements via a new relationship of utilization to specify the main parts of CAAs in the form of clear classes and contexts. In future work, the other UML diagrams will also be extended with concepts and vocabularies that highlight and ease the specification and documentation of the objects used in a CAS. Such extension must be consistent with the concepts developed in the proposed context-aware diagrams.

References

[1] Finkelstein, A. Savigni. "A Framework for Requirements Engineering for Context-Aware Service". In First International Workshop from Software Requirements to Architectures, 2011.

[2] Christof Simons, "CMP: A UML Context Modeling Profile for Mobile Distributed Systems". Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS'07), 2007.

[3] "(UML) Unified Modeling Language diagrams", available at [http://www.uml-diagrams.org/].

[4] 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.

[5] 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.

[6] G. Abowd and A. Dey. "Towards a better understanding of context and context-awareness". In Workshop on The What, Who, Where, When, and How of Context-Awareness (CHI2000), volume 1707 of Lecture Notes In Computer Science, pages 304–307. Springer, September 2000.

[7] Y. Oh, A. Schmidt and W. Woo. "Designing, Developing and Evaluating Context-Aware Systems". In Proceedings of the 2007 International Conference on Multimedia and Ubiquitous Engineering. IEEE Computer Society, 2007.

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

[9] K. Alghathber, C. Farkas and D. Wijesekera. "Securing UML information flow using flowUML". Research and Practice in Information Technology. 2006.

[10] G. Sindre and A. L. Opdahl. "Eliciting security requirements with misuse cases". Requirements Eng (2005) 10:34–44, DOI 10.1007/s00766-004-0194-4, London.

[11] G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper and M. Pinkerton. "A mobile context-aware tour guide". Wireless Networks, 1997.

[12] M. Bandinelli, F. Paganelli, G. Vannuccini and D. Giuli. "A context-aware security framework for next generation mobile networks". In Security and Privacy in Mobile Information and Communication Systems. Springer Berlin Heidelberg, 2009.

[13] M. Baldauf, S. Dustdar and F. Rosenberg. "A survey on context aware systems". International Journal of Ad Hoc and Ubiquitous Computing, 2(4):263{277, June 2007.

[14] 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.

[15] K. Peralta, P. Orozco, P. Zorzo, A. Oliveira. "Specifying security aspects in UML models". In 1st International Modeling Security Workshop. France, 2008.

[16] T. Xiaosheng, S. Qinghua and Z. Ping. "A Distributed Context-Aware Model for Pervasive Service Environment". IEEE Computer Society, 2006.

[17] M. Weiser. "Some Computer Science Issues in Ubiquitous Computing". ACM, 36:75-84, 1993.

[18] K. Henricksen and J. Indulska. "A software engineering framework for context-aware pervasive computing". In Proceedings 2nd IEEE Conference on Pervasive Computing and Communications, pages 77–86. IEEE Computer Society, Orlando, USA, 2004.

[19] J. Choi and Y. Lee. "Use-Case Driven Requirements Analysis for Context-Aware Systems". In: The Future Generation Information Technology Conference (Volume 353, pp 202-209), Korea, Springer, Heidelberg, 2012.

[20] 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.