2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT-IUCC-DASC-PICOM 2015)

An Extension of UML Activity Diagram to Model the Behaviour of Context-Aware Systems

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*—The main characteristics of context-aware systems are their ability to sense perpetually the context of the user and adapt their behaviour accordingly in response to changes in the user's context. Modularity is an important property for any design language as it provides ways of decomposing a system into smaller constituents which are then analyzed separately. The activity diagram of the Unified Modelling Language is a powerful language for describing the functions of a system. However it lacks notations for distinctively capturing the context-awareness requirements of context-aware systems. This paper proposes an extension of the activity diagram of Unified Modelling Language with new notations that enable the separation of concerns between the system functions, the context (or environment) constraints, and the adaption actions to be performed in reaction to changes in the system's environment. The pragmatics and flexibility of the proposed extension are demonstrated using a number of real-world case studies.**

Keywords—Software Requirement Engineering (SRE), Unified Modelling Language (UML), Context Aware System (CAS), Self-Adaptive System, Use Case Diagram and Activity Diagram.

I. INTRODUCTION

In this decade, many researchers are investigating the needs of self-adapting systems and their life-cycle development, which has faced many challenges. Context Source (CS) is the provider (the engine of information) which captures Context Information (CI) through a special transformation algorithm and carries it to such systems, as well as converting physical entities to virtual entities and inputting them into Context Aware System (CAS) as parameters [10]. Requirement specifications and modelling for CAS by the UML activity diagram are required in order to determine the required CI. Simulating the dynamic behaviour of CAS using new modelling approaches enhances CAS designers' awareness of the CAS behaviour and their integrations with all software and hardware [1 and 3].

The main problem for CAS is the changes in CIs, connection attributes, CS sensing and the user environments, which are in constant flux [1, 3 and 4]. Understanding system behaviour is a common challenge for many specialists in system modelling and simulation, especially with regard to complex systems such as CAS [12 and 19]. In addition, another issue that should be clarified is the interaction modelling between CAS and CS and how CI impacts CAS's behaviour and output [17, 18 and 20]. Furthermore, UML activity diagrams are still limited in depicting the behaviour of CASs and their components, processes and actions, such as the adaptation of output and acquisition senses [2]. Unlike traditional systems, CAS requires an advance level of analysis and modelling to achieve high performance response without delay to changing environmental context [13 and 14].

This paper proposes a new approach that extends UML activity diagrams with graphical notations to express the context-awareness aspects of CAS's behaviour. This novel extension is called a *context-aware activity diagram* and comprises three meta-swimlanes corresponding to the user activity, CAS activity and CS activity, respectively. This enables analysts to get a deep understanding of the CAS' behaviours and interactions with the users and CSs. The contributions of this work are threefold:

- New graphical notations are introduced to represent *context objects*, *context constraints* and *adaption activities*. Context objects store CIs sensed by CSs or aggregated from sensed data; context constraints express conditions upon context object attributes; and adaptation activities are activities that must be performed in response to specific changes in the system's context (Sect. III-A).
- A Novel notion of *context-aware activity diagram* is presented which extends the traditional UML activity diagrams to enable the representation of context objects, context constraints and adaptation activities. A typical context-aware activity diagram is partitioned into three meta-swimlanes describing the behaviours of the users, the CAS, and the CSs respectively. This approach results in a rich and expressive language for modelling the dynamic behaviour of CASs (Sect. III-B).
- The pragmatics of the proposed approach is evaluated using two real-world case studies (Sect. IV).

II. RELATED WORK

A. Traditional UML-Activity diagram

A UML Activity diagram models the dynamic aspects of a system, depicted as a flow of control from activity to activity.

An activity represents a task that the system must perform [2]. A typical UML activity diagram is shown in Fig. 1. Nodes are called *activity states* and the arcs between nodes are *transitions*. An activity state is rendered as rounded rectangle. The flow of control of an activity diagram begins in a *start state* rendered as a solid ball and terminates in an *end state* rendered as a sloid ball inside a circle. Swimlanes are used to partition the activity states between the actors responsible for executing them. A swimlane is rendered as a vertical solid line, while a transition is rendered as a directed line indicating that the flow of control passes from activity state to the next activity state. Special transitions called *fork* and *join* are used to synchronize parallel flows of control and are rendered as a thick horizontal line. A fork has one incoming transition and two or more outgoing transitions, each of which represents an independent flow of control. A join has two or more incoming transitions and one outgoing transition. Alternate transition sequences based on a Boolean expression can be specified using a *decision* transition rendered as a diamond. A decision has one incoming transition and two or more outgoing transitions; each outgoing transition carries a Boolean expression which must hold for that branch to be taken. These guards across outgoing transitions should not overlap and should cover all possibilities.

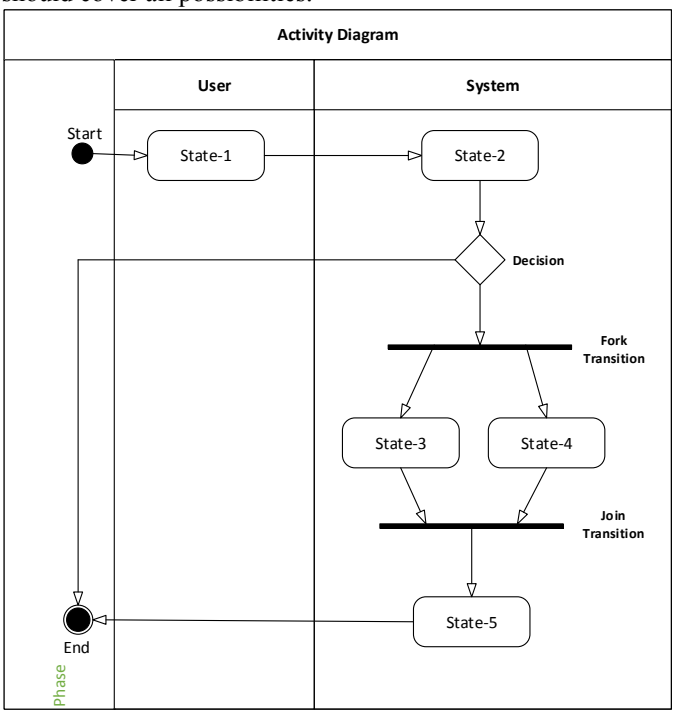


Figure 1 Illustration of an activity diagram

B. CAS Behaviour Modelling

System design, modelling and simulation are important phases before the actual implementation of any system. However system configuration is the last phase of system development and any flaw detected in this phase is costly, especially in complex systems [11 and 20]. Simulating different scenarios to produce models for CAS requirements will increase CAS quality and performance, and will also reduce various challenges when using CAS services, such as time and cost [21]. CAS modelling is still approached using different methodologies and there is a need for investigating several aspects, such as CAS run-time behaviour, CS networks and so on. Depictions of the behaviour of CAS, CS and the user environment are still poor and need more studies into the conversion of physical to virtual entities that are needed to support CAS functionalities, and also to express the interactions between the user, CAS and CS [7, 18, 19 and 20].

C. Existing approaches using UML-activity diagram

Researchers have recently developed new UML diagrams in their attempts to make the development of context-aware systems easier. Kang et al [5] and [6] extended the activity diagram to depict the requirements of mobile agent applications and their behaviour. Both papers investigated all aspects of agents and demonstrated the underlying computational models of mobile agent applications which used the upgraded version by UML 2.0 to design effective models. The authors used the activity diagram approach to model algorithmic behaviour captured for mobile agents, such as location, as well as to outline the various constraints which affect the mobile agents' behaviour and their communications.

Almutairi et al [7] enhanced the use case diagram to model the CAS. The authors created a section for the activity diagram and adjusted the new approach to include new notations for the activity diagram to model CAS, which focused on context categories, especially when and where to capture CI. Alghathbar et al [8] 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. G. Sindre and A. Opdahl [9] proposed a new concept of misuse cases based on the use case diagram. They focused on three techniques to identify and describe the misuse cases, specify the requirements needed by misuse cases at an early stage and test method to verify the security requirements.

III. A CONTEXT-AWARE ACTIVITY DIAGRAM

The advantages of UML over other system modelling languages are that it provides graphical notations for the visualisation, specification, construction and documentation of the artefacts of a software-intensive system. The activity diagram is one of the five UML diagrams for modelling the dynamic behaviours of traditional software systems. However, this diagram is too general to highlight specific characteristics of the behaviours of CASs such as context-awareness and dynamic adaptation. The context-aware activity diagram proposed in this section extends the traditional UML activity diagram with new graphical notations for visualizing, specifying, constructing and documenting the flows of control of CASs.

A. Modelling Context-awareness and Adaptation

In order to visualize and document the role of CIs in the behaviours of CASs, two new graphical notations have been introduced to represent a *context object* and a *context constraint*. Adaptation is modelled using special kind of activity state called *adaptation activity state* as depicted in

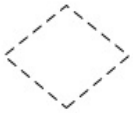
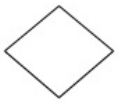
Fig. 2. A context object stores the value of a CI. It is rendered as a dashed rectangle and is connected to activity states using dependency relationships graphically represented as a directed dashed line. An incoming dependency indicates that the context object is an output; and an outgoing dependency means that the context object is an input to the activity state it is connected.

A context constraint models a decision making based on CIs acquired from CSs. It is rendered as a dashed diamond and has one incoming transition and two or more outgoing transitions. Each outgoing transition is labelled with a Boolean expression upon context object attributes that must hold for that transition to be taken. In general, each outgoing transition is connected to an adaptation activity state to be performed if the Boolean expression of that transition holds.

An adaptation activity state represents the activity to be performed in reaction to some changes in the environmental context. It is very important in CAS that these activities are done automatically when possible in order to minimise disruption to the user. An adaptation activity state is rendered as a dashed rounded rectangle as shown in Fig. 2.

Context-aware activity diagrams distinguish three categories of stakeholders: the users, the CAS, and the CSs. The activities and the interactions between these stakeholders make up the overall behaviour of a CAS. The notion of *meta-swimlane* is used to partition a context-aware activity diagram into three segments: the *user activity* segment, the *CAS activity* segment, and the *CS activity* segment. Each segment is the group of all the swimlanes associated with the corresponding category of stakeholder. The separation between consecutive meta-swimlanes is rendered as a vertical double solid line as depicted in Fig. 4. Each meta-swimlane can be further partitioned into many swimlanes, each describing the activities of an individual stakeholder within the corresponding category. Swimlanes are separated by a single solid line as shown in Fig. 5. The notations used to represent a context-aware activity diagram are summarized in Table 1.

Table 1: Context-aware activity diagram notations

Context related notations	UML notations
 Context Object	 Object
 Context Constraint	 Decision
 Adaptation Action	 State
 Meta-swimlane separation	Swimlane separation

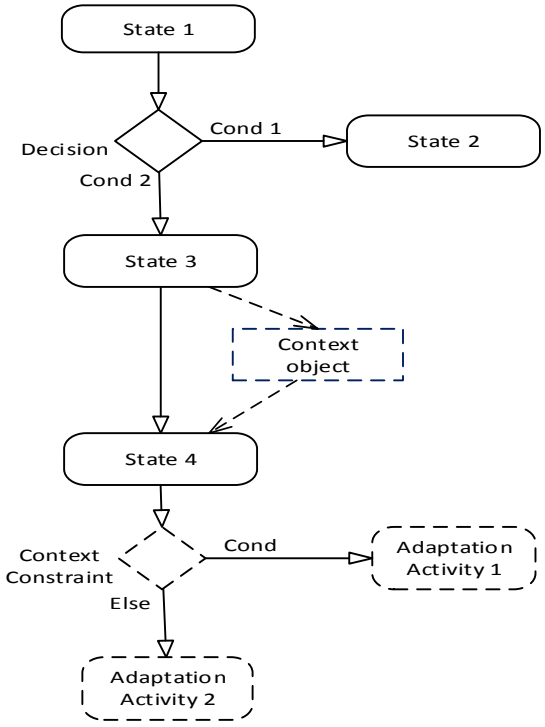


Figure 2: Illustration of a context-aware activity diagram

B. Context Acquisition

Context acquisition determines the correct CSs (such as sensors) to specify suitable software for the CS and ensure that data output is provided by the CS to identify the task queries and how they can manage and save changed CIs (as shown in Fig. 3). A CI is any physical, tangible object, and usually the main task of sensor devices (context service) is to capture specific CI and carry it to applications for different uses, such as location, vehicles, buildings, human beings, and devices. Delivery requirements specify different methods that enhance the understanding of how CSs integrate with each other as well as how to send, receive and analyze CI. Tracking the user and updating his or her location requires different context services, and problems can occur with such providers, resulting in disconnection. For example, sensors are dependent on their devices and services can capture the CI using special coding. There are many types of sensors and devices, such as physical, virtual and logical sensor services, as well as specific sensors created to capture different kinds of CI, such as temperature, speed, movement, photo, light and so on.

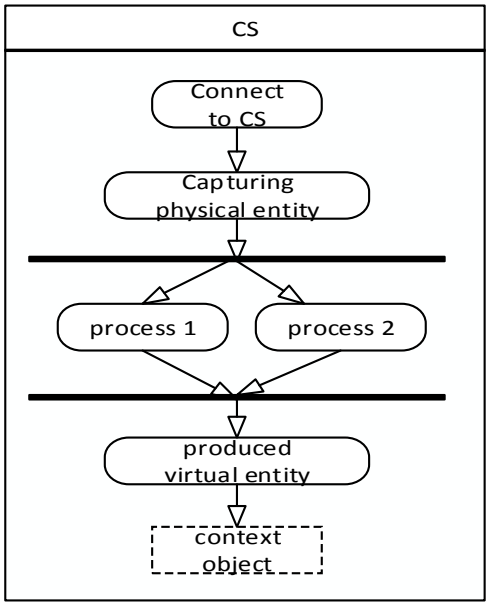


Figure 3: An illustration of CS activities

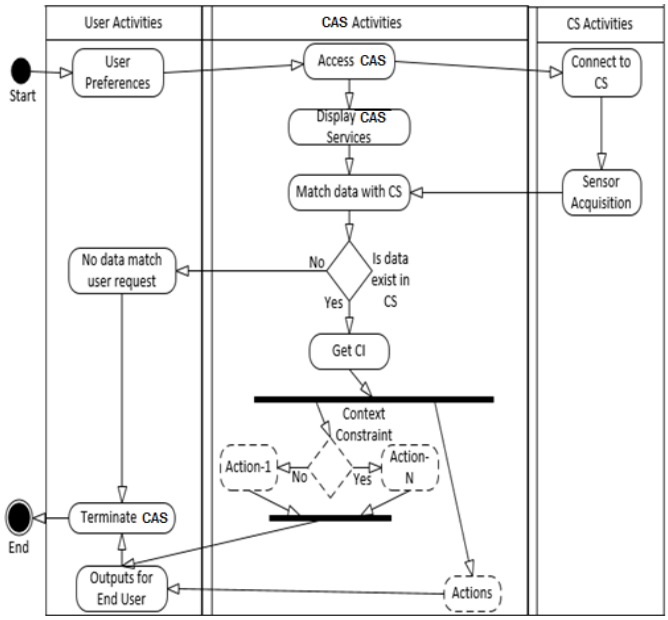


Figure 4: A framework of high-level interactions

C. Frameworks for context-aware activity diagram

Standard frameworks for context-aware activity diagram and their notations investigate the dynamic behaviour for CAS modelling. This serves to simplify complicated issues regarding CAS behaviour and interactions with other components step by step, by using current activity diagram notations and new notations of context activity diagram. An approach of context-aware activity diagram shows the sequence and flow of actions and CIs activities performed by CASs; it also illustrates how the activity model of CAS will be controlled to meet the requirements specified in the use case model. A context-aware activity diagram can be used to describe high-level interactions (which illustrate the tracing activities for the main components without expressing other details for each component) and low-level interactions (which demonstrate more interaction activities within components).

A framework of high-level interactions graphically represented as meta-swimlanes showing the main components of the system. It outlines three meta-swimlanes, namely the user profile meta-swimlane, the CAS meta-swimlane and the CS meta-swimlane. By contrast, a framework of low-level interactions graphically represented as swimlanes depicts the interaction activities between components within individual meta-swimlanes. This separation of concerns help to ensure that CAS activity models are well detailed and easily understood in order for designers to translate CAS models into source code with greater confidence. The following section of the case studies investigates the proposed frameworks and when each framework should be used. The framework of high-level interactions is shown in Fig. 4. The framework of low-level interactions is depicted in Fig. 5, showing detailed interactions between the stakeholders across swimlanes. The three meta-swimlanes classify the dynamic modelling of CAS into three kinds of activity as follows:

- *User Activity*: The user meta-swimlane shows the user activities such as login, user preferences and security activities, which represent the sequence and flow of actions to be performed by CAS security to check the user's account validity and authentication as well as to provide options such as whether or not to remain logged on for a later use.
- *CAS Activity*: The objective of this meta-swimlane is to model CAS behaviour and their adaptation activities. The differences between traditional systems and CAS interact (implicitly or explicitly) with the users and CSs to gather CIs which are used to make adaptation decisions.
- *CS Activity*: The main function of CS is to acquire CI by sensing specific aspects of the environment context. In addition, there are numerous processes relating to context acquisition, including starting the sensors and reading the different objects required to create CIs to be sent to the CS Server and saved in the CS databases. In addition, modelling CS activities is important for several reasons; it enables CAS designers to know which CS will interact with CAS, understand CI parameters and know CS database schema.

A framework of low-level interactions provides a set number of swimlanes for the user meta-swimlane, the CAS meta-swimlane and the CS meta-swimlane as follows (see Fig. 5):

- User swim-lanes: user-1, user-2, …, user-N
- CAS swim-lanes: sub-CAS-1, sub-CAS-2, …, sub-CAS-M
- CS swim-lanes: CS-1, CS-2, …, CS-K.

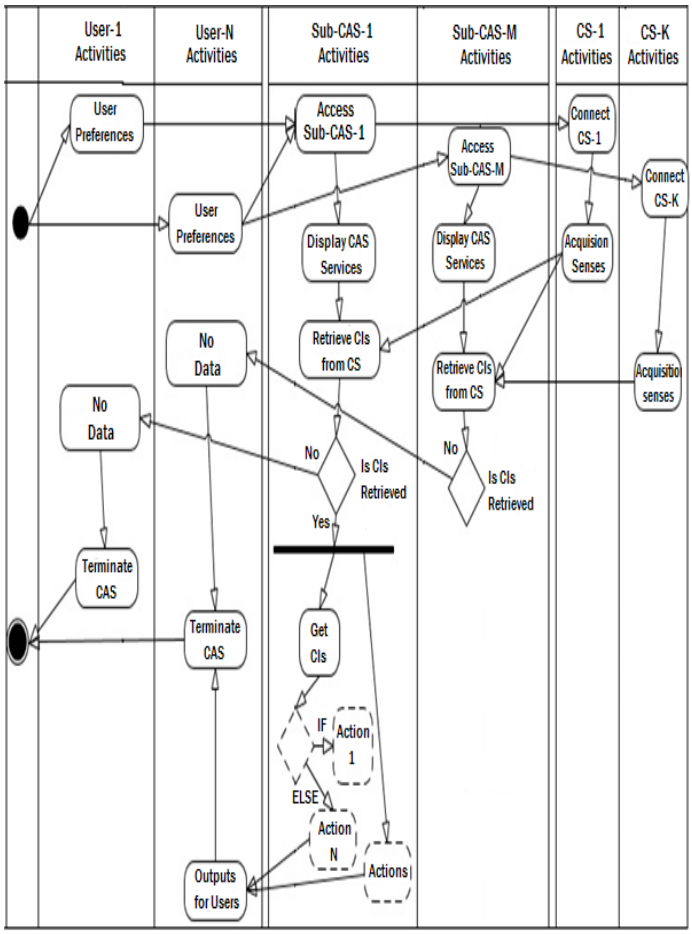


Figure 5: A framework of low-level interactions

IV. 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 activity diagram for a temperature control system; and the second one presents a context-aware activity diagram for a navigation system.

A. A Teamperature Control System

Services of Temperature Control System (TCS) are important for many reasons: to ensure users' comfort, to save energy and to act as self-controllers in cases when TCS work as a self-adapting system, responding to the surrounding temperature. The main algorithm used for this system is an On/Off controller depending on a temperature set-point specified by the user and the output action is applied when the temperature crosses the user's set-point [13]. The On/Off controller is a special algorithm used to switch heating on or off to create warming, regulating or cooling actions depending on the surrounding temperature when sensed by special sensors (such as RTD or a Thermocouple). Any system needs to go through a series of different stages before implementation to ensure that it is accurate and is meeting the correct objective, for which the user requirements must be specified carefully. The user requirements in this case study are to use a self-adapting TCS to enable the user to set preferences and turn on the system without checking the room temperature to change the Air-Conditioning mode depending on the room temperature. In this case study (as depicted in Fig. 6 and summarized in Table 2) the temperature range classification depends on preferences, which are defined as follows:

- TCS applies Heating mode when the temperature of the user's room is less than or equal to 15 degrees
- TCS applies Warm mode when the temperature of the user's room is more than 15 degrees but less than or equal to 25 degrees
- TCS applies Air-Fan mode when the temperature of the user's room is more than 25 degrees and less than or equal to 37 degrees
- TCS applies Cooling mode when the temperature of the user's room is more than 37 degrees.

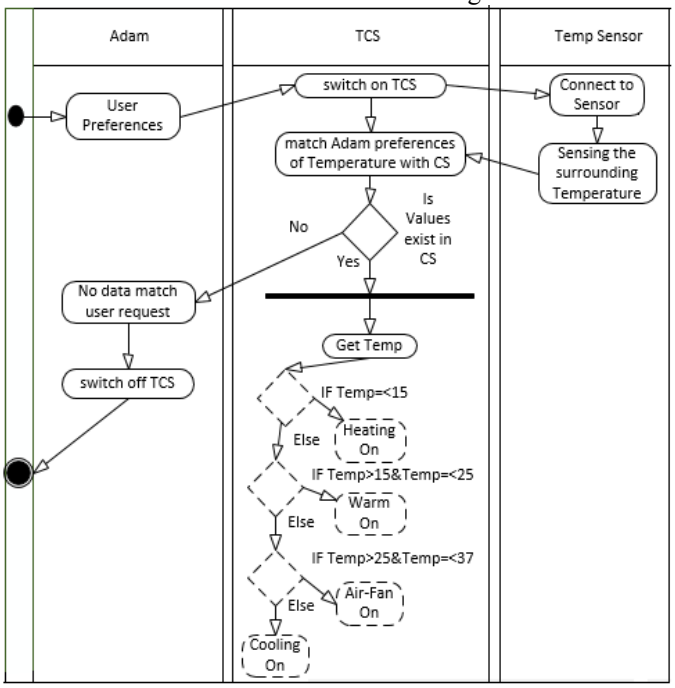


Figure 6: The main activities of a Temperature Control System

Table2 : Summary of TCS behaviour activities

TCS	Adam
CAS Adaptation Activity	Heating, warm, air-fan and cooling depend on temperature
CS Activity	Temperature sensor produces surrounding temperature for the user
CI Type	Atomic CI (Temperature)
CAS Constraint	- Heating if equal to or less than 15 - Warm if equal to or less than 25 - Air-Fan if equal to or less than 37 - Cooling if equal to or more than 37

B. A Navigation System

Services of Navigation System (NS) include two sub-CASs: Navigator sub-CAS and Traffic sub-CAS; NS provides important information for users, specifying the user's start point, destination point and journey path using the most convenient route, and the remaining time for the journey. However NS provides important services to guide and direct users without errors or delays [14]. The use of smart devices by many users around the world and the fact that transportation has become faster and more comfortable means that it is easier than ever to reach one's required destination.

The main CS in NS is GPS, used to produce required CIs for NS and also to monitor changes, which are carried into different applications. In addition GPS is popular CS and the main CI provided by GPS is location, which includes the X and Y coordinates. X and Y coordinates are used around the world to specify a unique serial number for every location on Earth. In addition NS has a map application that can be used to search for information about an address or location; the device system services also functions to direct the user to his or her required destination. However, we need to identify the difference between CAS services and information. The user does not always use the application services and may only occasionally require information without application services. This case study depicts NS behaviour and classifies it into three meta-swimlanes for the user, the CAS and the CS (as shown in Fig. 7).

The user swim-lane includes swimlanes for two user activities which require a secure login for Adam only. Security activities within the user swimlane include three functionalities required for CAS to verify the user authentication: the first activity (Login) is to verify that the user name and password are correct, while the second activity is to check the account's validity to access CAS services, such as whether the account needs to be renewed to allow longer or unlimited access depending on the user's contract with the CAS provider. The third activity is to check whether the user would like CAS to remember his account login details to allow him to access CAS again later without having to re-enter his user name and password, although the user can decline this option for greater privacy. In addition, the user's position and destination are the most important options used as settings in the GPS system to specify the user preferences for the location details that are required to inform the navigator. The most important data is usually the post code, which is required by Navigator System databases as the primary key for retrieval purposes to find the full details, such as position details, coordinates and which direction the destination is in.

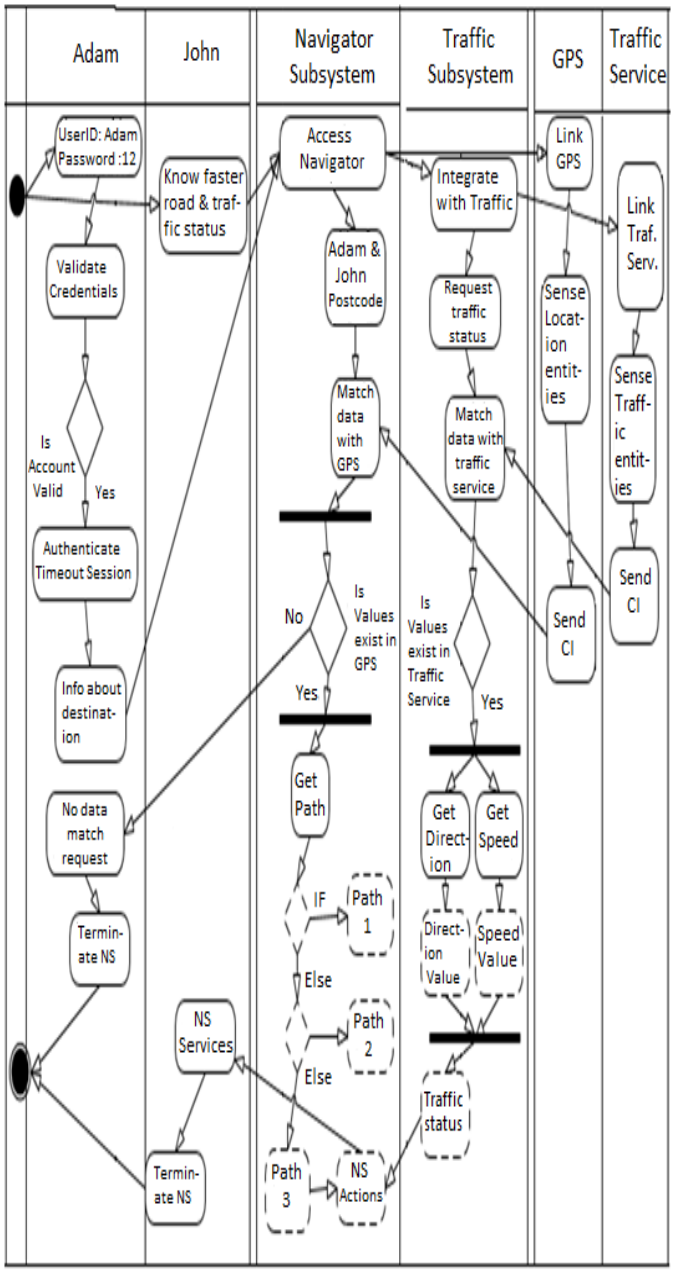


Figure 7: The main activities of a Navigation System

V. CONCLUSION

This paper presented a new extension of the activity diagram of UML to cater the behaviour activities of CAS, CS. This novel extension is called a *context-aware activity diagram*. The innovative features of a context-aware activity diagram includes (i) a new notation for context object that distinguishes them from other types of object in the system; (ii) a new graphical notation for modelling context constraints that are used in adaptation decision making; (iii) a new notion of adaptation activity state for modelling the adaptive behaviour of a CAS; (iv) a concept of meta-swimlane is introduced to partition a context-aware activity diagram between three stakeholder categories: the user, the CAS and

the CS. This results in a rich and expressive graphical language for describing the behaviours of CASs.

In future work, the UML class diagram 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 activity diagram.

REFERENCES

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

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

[3] 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.

[4] 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.

[5] M. Kang and K. Taguchi."Modelling Mobile Agent Applications by Extended UML Activity Diagram". In: 6th International Conference on Enterprise Information Systems (ICEIS)' 04. Porto, Portugal.2004.

[6] M. Kang, L. Wang and K. Taguchi. Modelling Mobile Agent Applications in UML 2.0 Activity Diagrams. In: 3rd SELMAS Workshop at ICSE 104-111. 2004.

[7] 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.

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

[9] 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.

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

[11] 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.

[12] J. Jimenez and L. Iribarne. "Describing use-case relationships with sequence diagrams". The Computer Journal, 2007.

[13] G. Anne Ngu. "Context aware actors". In Presented at the Ninth Biennial Ptolemy Miniconference, Berkeley, 2011.

[14] 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.

[15] P. Singhala, D. Shah, B. Patel, "Temperature Control using Fuzzy Logic", International Journal of Instrumentation and Control Systems (IJICS) Vol.4, No.1, India, January 2014.

[16] 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.

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

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

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

[20] 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.