

Except as otherwise permitted under the Copyright, Designs and Patents Act 1988, this thesis may only be produced, stored or transmitted in any form or by any means with the prior permission in writing of the author. The author asserts his/her right to be identified as such in accordance with the terms of the Copyright, Designs and Patents Act 1988.

Requirements Engineering of Context-Aware Applications

Ph.D. Thesis

Ahmed Mohammed Alalshuhai

This thesis is submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Software Technology Research Laboratory

De Montfort University

Leicester – United Kingdom

2015

Dedication

To my parents and brothers

This thesis is dedicated to my Father, Mohammed Alalshuhai, who was the motivation behind this thesis, and to all my family members, especially my Mother.

It is also dedicated to my wife, sons and daughter and to all those who believe in me, without whose support during the hardest of times, this work would not have been possible.

Abstract

Context-aware computing envisions a new generation of smart applications that have the ability to perpetually sense the user's context and use these data to make adaptation decision in response to changes in the user's context so as to provide timely and personalized services anytime and anywhere. Unlike the traditional distribution systems where the network topology is fixed and wired, context-aware computing systems are mostly based on wireless communication due to the mobility of the network nodes; hence the network topology is not fixed but changes dynamically in an unpredictable manner as nodes join and the leave network, in addition to the fact that wireless communication is unstable. These factors make the design and development of context-aware computing systems much more challenging, as the system requirements change depending on the context of use.

The Unified Modelling Language (UML) is a graphical language commonly used to specify, visualize, construct, and document the artefacts of software-intensive systems. However, UML is an all-purpose modelling language and does not have notations to distinguish context-awareness requirements from other system requirements. This is critical for the specification, visualization, construction and documentation of context-aware computing systems because context-awareness requirements are highly important in these systems. This thesis proposes an extension of UML diagrams to cater for the specification, visualization, construction and documentation of context-aware computing systems where new notations are introduced to model context-awareness requirements distinctively from other system requirements. The contributions of this work can be summarized as follows:

- (i) A context-aware use case diagram is a new notion which merges into a single diagram the traditional use case diagram (that describes the functions of an application) and the use context diagram, which specifies the context information upon which the behaviours of these functions depend.
- (ii) A Novel notion known as a 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. Context constraints express conditions upon context object attributes that trigger adaptation activities; adaptation activities are activities that must be performed in response to specific changes in the system's context. (iii) A novel notion known as the context-aware class diagram is presented, which extends the traditional UML class diagrams to enable the representation of context information that affect the behaviours of a class. A new relationship, called utilisation, between a UML class and a context class is used to model context objects; meaning that the behaviours of the UML class depend upon the context information 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 contexts upon which they depend. The pragmatics of the proposed approach are demonstrated using two real-world case studies.

Declaration

I declare that the work described in this thesis is original work undertaken by me for the degree of Doctor of Philosophy, at the software Technology Research Laboratory (STRL) at De Montfort University, United Kingdom.

No part of the material described in this thesis has been submitted for any award of any other degree or qualification in this or any other university or college of advanced education.

Ahmed Mohammed Alalshuhai

Publications

1. A. Al-alshuhai and F. Siewe. "An Extension of the Use Case Diagram to Model Context-Aware Applications". Intelligent Systems Conference (SAI-2015), London, UK, November 10-11, 2015.
2. A. Al-alshuhai and F. Siewe. "An Extension of the UML Activity Diagram to Model the Behaviour of Context Aware Systems". The 15th IEEE International Conference on Computer and Information Technology (CIT-2015), Liverpool, UK, October 26-28, 2015.
3. A. Al-alshuhai and F. Siewe. "An Extension of the Class Diagram to Model the Structure of Context-Aware Systems". The Sixth International Joint Conference on Advances in Engineering and Technology (AET-2015), NCR-Delhi region, India, December 26, 2015.
4. F. Siewe and A. Al-alshuhai. "From Use Case Diagrams to Executable Context-aware Ambients". The 10th International Conference for Internet Technology and Secured Transactions (ICITST-2015), London, UK, December 14-16, 2015.
5. F. Siewe, A. Al-alshuhai, S. Almutairi and A. Almutairi. "Analysing Use Case Diagrams in a Calculus of Context-aware Ambients". International Journal of Intelligent Computing Research, 2016.

Acknowledgements

First and foremost, my truthful thankfulness goes to the most merciful **ALLAH** for all things he blessed me with throughout my whole life. Without those blessings, I would not be in this position at all.

This thesis would not have been successful without the support of my supervisor **Dr Francois Siewe**, whom I would like to thank for his professional guidance, critical comments and technical suggestions. He always took time to evaluate this thesis, from the start of this research until the last day, even during weekends and holidays.

I would like to express my endless thanks to **Dr Helge Janicke**, the head of STRL for his support, help and guidance. Also thanks to my second supervisor **Dr Feng Chen**, who was helpful and offered assistance, and to all STRL members.

I want to thank my father for his support and trust in me. After studying three degrees – bachelor's, master's and PhD – I gained the most important thing: an appetite for researching and writing, developing new approaches and ideas and exchanging them with good friends around the world to make even the hardest of times enjoyable.

Also I would like to thank my mother for her endless support and prayers since the day I was born. Moreover, all appreciation goes to my brothers for their continuous support and for sharing the happy and sad times.

Finally I would like to thank my wife for her endless love and prayers; I appreciated her care for our children during the hardest times. Also I would like to acknowledge my sons and daughter.

List of Figures

2.1 Iterative processes of requirements engineering.....	22
2.2 Iterative processes of system modelling, design and simulation.....	27
2.3 CAA Behaviour Simulation.....	28
2.4 CAA Structure Simulation.....	29
2.5 UML Categories.....	30
2.6 Use case diagram notations.....	33
2.7 Activity diagram notations.....	39
2.8 Class diagram shape.....	40
2.9 An example diagram of the Choi approach.....	46
2.10 An example diagram of the Almutairi approach.....	47
3.1 Traditional use case diagram notations and their relationships.....	55
3.2 An example of adjustable use case diagram notations.....	59
3.3.A An illustration of the notations for use context diagrams.....	64

3.3.B An example of a context association relationship.....	65
3.3.C An example of an “include” relationship.....	66
3.3.D An example of an “extend” relationship.....	66
3.4 An illustration of the notations for a context-aware use case diagram.....	73
3.5.A An example of one-to-one capturing method.....	74
3.5.B An example of one-to-many capturing method.....	74
3.5.C An example of many-to-one capturing method.....	75
3.6 A standard modelling of a context-aware use case diagram for CAAs.....	76
3.7 Driving control system functionalities.....	80
4.1 Activity diagram swimlanes and notations.....	85
4.2 An example of adjustable notations of the activity diagram.....	89
4.3 An example of the notations of the context activity diagram.....	93
4.4.A Context-aware activity diagram notations to show the activities of context constraint.....	94
4.4.B Context-aware activity diagram notations to show the activities of	

adaptation action.....	95
4.5.A An illustration of CS activities.....	97
4.5.B An illustration of flow-activities between CAA and CS.....	98
4.6 An illustration of adaptation actions and their constraints.....	99
4.7.A Context-aware activity diagram modelling of high-level interactions.....	103
4.7.B Context-aware activity diagram modelling of low-level interactions.....	105
4.8 Context-aware activity diagram activities of temperature-control system using high-level interactions.....	108
5.1 The partitions of class shape.....	114
5.2 An example of the adjustable elements of the class diagram.....	119
5.3 An illustration of shape partitions for the context class diagram.....	120
5.4 An illustration of elements for the context aware class diagram.....	122
5.5 The structure classes of CAAs' components.....	124
5.6 The structure contexts of CAAs' components.....	126
5.7 A standard modelling of a context aware class diagram.....	128

5.8 Temperature control system structure.....	129
6.1 Context-aware navigation system Infrastructure.....	134
6.2.A Functional requirement of user preferences for navigation subsystem.....	137
6.2.B Functional requirement of know faster path for navigation subsystem.....	138
6.2.C Functional requirements of know remaining time and speed for navigation subsystem.....	138
6.2.D Functional requirement of Get position for navigation subsystem.....	139
6.3.A Context awareness requirements of distance and speed.....	140
6.3.B Context awareness requirements of map data and coordinates.....	140
6.3.C Context awareness requirements of speed limits, time to destination and coordinates for navigation subsystem.....	141
6.4 A navigation subsystem.....	143
6.5.A An example to show the notations of the adjustable activity diagram.....	148
6.5.B An example to show the notations of the context activity diagram.....	149
6.5.C The main activities of the navigation and traffic subsystems.....	151

6.5.D An illustration of remaining time activities.....	152
6.6.A An example to show the elements of the adjustable class diagram.....	155
6.6.B An example to show the elements of the context class diagram.....	156
6.6.C The static structure of the navigation subsystem.....	157
6.7.A Functional requirement of user preferences for the traffic subsystem.....	160
6.7.B Functional requirements of traffic information and know speed limit.....	161
6.7.C Functional requirement of alternative direction.....	161
6.8.A Context awareness requirements of direction and speed.....	162
6.8.B Context awareness requirements of direction and movement.....	163
6.8.C Context awareness requirement of speed for traffic subsystem.....	163
6.9 A traffic subsystem.....	164
6.10.A An example to show the notations of the adjustable activity diagram.....	169
6.10.B An example to show the notations of the context activity diagram.....	170
6.10.C The main activities of the traffic subsystem.....	172

6.11.A An example of class diagram elements for the traffic subsystem.....	173
6.11.B An example of context class diagram elements for the traffic subsystem...	175
6.11.C The static structure of the traffic subsystem.....	176
7.1 Context-aware weather forecast system Infrastructure.....	181
7.2.A Functional requirement of user preferences for weather forecast system.....	185
7.2.B Functional requirement of display weather forecast for weather forecast system.....	186
7.3.A Context awareness requirements of location, temperature and rain for weather forecast system.....	187
7.3.B Context awareness requirements of snow and humidity for weather forecast system.....	187
7.4 A weather forecast system.....	188
7.5.A An example of weather forecast system and CS activities using an activity diagram.....	197
7.5.B An example of adaptation actions for weather forecast system using context activity diagram.....	198
7.5.C Weather forecast system behaviour modelling using a context-aware	

activity diagram.....	199
7.6.A An example of the weather forecast system using an adjustable class diagram.....	201
7.6.B An example of the weather forecast system using a context class diagram...	202
7.6.C The structure components of the weather forecast system.....	203

List of Tables

2.1 Examples of project failure.....	18
2.2 Requirement types.....	20
2.3 Comparison of UML diagrams.....	32
2.4 Usage scenario template.....	36
2.5 Context categories.....	43
2.6 Comparison of six approaches.....	45
3.1 Comparison between traditional use case diagram, adjustable use case diagram and use context diagrams.....	58
3.2 Adjustable notations of use case diagram for CAAs.....	61
3.3 New notations of use context diagram for CAAs.....	62
3.4 An example of a context aware usage scenario template.....	68
3.5 Notations for describing a context-aware use case diagram.....	72
4.1 Comparison between traditional activity diagram, adjustable activity diagram and context activity diagram.....	88

4.2 Adjustable notations of activity diagram for CAAs.....	90
4.3 New notations of context activity diagram for CAAs.....	91
4.4 Notations for describing a context-aware activity diagram.....	96
4.5 Summary of temperature-control system behaviour activities by context-aware activity diagram.....	107
5.1 Comparison between traditional class diagram, adjustable class diagram and context class diagram.....	117
5.2 Adjustable elements of class diagram for CAAs.....	119
5.3 Elements for describing a context-aware class diagram.....	123
6.1 Summary of functional requirements and context awareness requirements for navigation subsystem.....	142
6.2 Functional requirement and context awareness requirement of navigation subsystem using context-aware usage scenario template.....	144
6.3 Summary of behaviour activities for navigation subsystem.....	153
6.4 Summary of functional requirements and context awareness requirements for traffic subsystem.....	164
6.5 Functional requirement and context awareness requirement of traffic	

subsystem using context-aware usage scenario template.....	165
6.6 Summary of behaviour activities for the traffic subsystem.....	171
7.1 Summary of functional requirements and context awareness requirements for weather forecast system.....	189
7.2 Functional requirement and context awareness requirement of weather forecast system using context-aware usage scenario template.....	191

List of Abbreviations

CAA	Context Aware Application
CANS	Context Aware Navigation System
CAS	Context Aware System
CASE	Computer-Assisted Software Engineering
CASLC	Context Aware System Life Cycle
CA-UML	Context Aware Unified Modelling Language
CAWA	Context Aware Web-Application
CAWFS	Context Aware Weather Forecast System
CA-XML	Context Aware Extensible Markup Language
CCA	Calculus of Context-aware Ambients
CI	Context Information
CS	Context Source
DB	Data Base
GPS	Global Positioning System

ICT	Information and Communications Technology
IPO	Input-Process-Output
IS	Information System
IT	Information Technology
OCL	Object Constraint Language
OMG	Object Management Group
OOA	Object Oriented Analysis
SDLC	Software Development Life-Cycle
SE	Software Engineering
SHS	Smart Home System
Ubicomp	Ubiquitous computing
UML	Unified Modelling Language
PDA	Personal Digital Assistant
PC	Personal Computer
XML	Extensible Markup Language

Contents

Dedication.....	III
Abstract.....	IV
Declaration.....	VI
Publications.....	VII
Acknowledgements	VIII
List of Figures.....	IX
List of Tables	XVI
List of Abbreviations	XIX
Contents	XXI
1. Introduction.....	1
1.1 Research Background.....	1
1.2 Motivation	3
1.3 Thesis Scope and Research Questions	4
1.4 Research Methodology.....	6
1.5 Thesis Contribution	8
1.6 Measures of Success.....	10
1.7 CA-UML Diagrams.....	11
1.8 Thesis Outline.....	13
2. Literature Review	15
2.1 Introduction	15
2.2 Requirement Engineering and Modelling	16
2.2.1 Requirements Gathering and Analysis.....	19
2.2.2 Requirements Engineering Processes.....	22
2.2.3 Requirements Categories.....	24
2.3 System Modelling, Design and Simulation.....	27
2.3.1 CAA Behaviour Simulations.....	28
2.3.2 CAA Structure Simulations.....	29
2.4 UML	30
2.4.1 Use Case Diagram.....	33

2.4.2 Usage Scenario Template.....	34
2.4.3 Activity Diagram.....	38
2.4.4 Class Diagram	40
2.4.5 Stereotypes	40
2.5 Context Aware System.....	41
2.5.1 Context	43
2.5.2 Context Source	44
2.6 Existing UML approaches for requirements modelling of CAA and their limitations.....	45
2.7 Summary	51
3. An Extension of the Use Case Diagram to Model Context Aware Applications	53
3.1 Introduction	53
3.2 Traditional Use Case diagram	55
3.3 CAAs Functionalities Modelling	58
3.3.1 Adjusting the notations of the Use Case diagram for CAAs modelling.....	59
3.3.2 Creating the notations for the Use Context diagram for CAAs modelling	62
3.3.3 Context Aware Usage Scenario Template	67
3.4 A Context Aware Use-case Diagram	70
3.4.1 Applicable capturing methods by Context Source	73
3.4.2 A standard Context Aware Use-case diagram.....	76
3.4.3 Specifying the functional requirements for CAAs using a Use Case diagram.....	77
3.4.4 Specifying the context-awareness requirements for CAAs using a Use Context diagram.....	78
3.4.5 A practical example of a driving control system using a context-aware use case diagram.....	79
3.5 Summary	80
4. An Extension of the Activity Diagram to Model the Behaviour of Context Aware Applications	82
4.1 Introduction	82
4.2 Traditional Activity diagrams	85
4.3 CAAs Behaviour Modelling.....	87
4.3.1 Adjusting the notations of the Activity diagram for CAAs Modelling	88
4.3.2 Creating the notations of Context Activity diagram for CAAs Modelling	90
4.4 A Context Aware Activity Diagram.....	93
4.4.1 Specifying context acquisition of CS behaviour using a context-aware activity	

diagram.....	96
4.4.2 Specifying the behaviour of Adaptation actions using a context-aware activity diagram.....	98
4.4.3 Standard Context Aware Activity diagrams.....	99
4.4.4 A practical example of a Temperature Control System using a context-aware activity diagram.....	106
4.5 Summary	109
5. An Extension of the Class Diagram to Model the Structure of Context Aware Applications	111
5.1 Introduction	111
5.2 Traditional class diagrams.....	114
5.3 CAAs Structure Modelling.....	116
5.3.1 Adjusting the elements of the Class diagram for CAA modelling.....	118
5.3.2 Creating the elements of the Context Class diagram for CAAs modelling.....	120
5.4 A Context-Aware Class Diagram.....	121
5.4.1 Specifying the structure classes of CAAs' components using the context-aware class diagram.....	123
5.4.2 Specifying the structure contexts of CAAs components using a context-aware class diagram.....	125
5.4.3 A standard Context Aware Class diagram	126
5.4.4 A practical example of a temperature control system using context-aware class diagram.....	129
5.5 Summary	130
6. Real-life Case Study of a Context-Aware Navigation System using CA-UML diagrams	132
6.1 Introduction	132
6.2 The infrastructure of a Context-Aware Navigation System.....	133
6.3 Modelling the Navigation Subsystem using CA-UML diagrams	135
6.3.1 Specifying the functionalities of the Navigation Subsystem using the context-aware use case diagram notion	136
6.3.2 Modelling the dynamic behaviour of the Navigation Subsystem using the context-aware activity diagram notion	147
6.3.3 Designing the static structure of the Navigation Subsystem using the context-aware class diagram notion.....	154
6.4 Modelling the Traffic Subsystem using CA-UML diagrams.....	159
6.4.1 Specifying the functionalities of the Traffic Subsystem using the context-aware use case diagram notion.....	159
6.4.2 Modelling the dynamic behaviour of Traffic Subsystems using the context-aware	

activity diagram notion	168
6.4.3 Designing the static structure of the Traffic Subsystem using the context-aware class diagram notion	173
6.5 Summary	176
7. Real-life Case Study of a Context-Aware Weather Forecast System using CA-UML diagrams	179
7.1 Introduction	179
7.2 The infrastructure of a Context-Aware Weather Forecast System.....	181
7.3 Modelling a Weather Forecast System using CA-UML diagrams.....	183
7.3.1 Specifying the functionalities of a Weather Forecast System using the context-aware use case diagram notion	184
7.3.2 Modelling the dynamic behaviour of a Weather Forecast System using the context-aware activity diagram notion	196
7.3.3 Designing the static structure of a Weather Forecast System using the notion of the context-aware class diagram	200
7.4 Summary	204
8. Conclusion and Future Work.....	207
8.1 Research Summary.....	207
8.2 Statement of Evaluation	211
8.3 Research Questions Revisited	213
8.4 Contribution to Knowledge	214
8.5 Future Work	216
Bibliography	220

1. Introduction

Objectives:

- Present an introduction and outline the motivation for this research;
- List the research question and sub-questions;
- Specify the scope of this thesis;
- Present the research methodology and contribution;
- Define the CA-UML and their standard modelling notions for CAAs;
- Outline the thesis structure.

1.1 Research Background

Ubiquitous computing envisions a new generation of intelligent applications that have the ability to perpetually gather data about the user's context and use these data to make adaptations to services in response to changes in the user's environment. Such applications generally run on a mobile device carried by the user and use a variety of sensors to gather context data [1, 5, 13, 14].

Today, context-awareness requirements have become necessary to enable next generation technology to fulfil the requirements of intelligent systems, which need to sense different objects and their environments and use the extracted contexts to adapt their behaviour [8, 24]. Furthermore, in context-aware applications, context can be classified into three types:

- The user context or personal context, which includes, for example, the user's id, preferences, and personal health information.
- The device context or ICT context, which encompasses the user's mobile device capabilities, network connectivity, and battery power.
- The physical environment context, such as time, light, location and weather.

CHAPTER 1. INTRODUCTION

Various types of sensor are used to measure context information: these may be physical sensing devices, such as a GPS (to sense the user's location), an accelerometer (to sense the user's movement) or a temperature sensor; or they may be virtual sensors like the user's calendar (to sense the location or the activity of the user) or a weather web service. The generic term 'context source' is used in this thesis to refer to sensors, whether physical or virtual, while 'context information' refers to the sensed data. A context-aware application collects context information via context sources and uses this information to adapt its behaviour so as to assist the user with relevant information and services at any time. The behaviour of such an application is context dependent [25, 26, 29].

As a result, context information and context source play an important role in the requirement analysis of context-aware applications. In software engineering, UML diagrams are commonly used to conceptualize the functions and services of software applications [6, 9].

Recently, CAAs have been making major technological advances with smart devices that help users to perform many business processes by using integrated applications. These applications need distributed databases within CAA interactions via ad-hoc networks: this approach is a popular method and uses special functions and mathematical variables to consider an active query for examination. This approach is also deployed faster and supports any time and location to obtain the required information, such as the ACQUIRE approach [11].

Additionally, global variables provide web services which include shared functions for special coding and save extracted context information. This study sets out to develop UML extension of use case diagrams, activity diagrams and class diagrams, and to discuss existing approaches for requirements engineering of CAAs and their weaknesses in adapting to context information changes. Finally, this chapter reports different aspects of this research, such as the study's motivation, the scope of the thesis and research questions, methodology, contribution, measures of success and the thesis structure in the following sections.

1.2 Motivation

This section explains the motivation for CAA modelling by describing the advantages for the static and dynamic modelling of CAAs, and also explains the emerging importance of CAAs and the issues associated with software engineering such as modelling and design. This research selected the most widely used UML diagrams of use case, activity and class diagrams to provide special modelling aspects to model system architecture, which express the system functions, objects and services in the modelling stage of SDLC.

CAAs have attracted increasing research attention in this decade, along with understanding and using context information and their related components, such as context service or CAA clients. The main objective in this research is to evaluate UML diagrams for CAA modelling and simulation, which calls for investigation of a range of approaches for all aspects of CAAs to create thorough modelling approaches. This thesis suggests a new approach called CA-UML diagrams to visualize, analyse, document and design the scope of CAAs and their services. In addition, this thesis is motivated by the need for modelling using the CA-UML approach to analyse the interactions of CAA and context source, which calls for capturing of contexts depending on specific requirements and needs. The main advantage is to provide support for CAA designers by creating new approaches to enhance CAA models to express the exact requirements and CAA functions. Furthermore, this thesis suggests set concepts for CA-UML diagrams to describe the different aspects of CAAs' modelling using the notions of context aware use-case diagrams, context aware activity diagrams and context aware class diagrams. The main objective of this thesis is to depict the CAA to provide details at high and low levels using various techniques. The main contributions of this thesis are in abstracting the functionalities, behaviour and structure of CAAs. Most research approaches and CAA developers attempt to develop different solutions to produce benefits for the client, to match users' requirements and to resolve their circumstances. With regard to this goal, it is necessary to investigate the context information required in order for CAA and context source to work together and use special methods to collect contexts and interpret this information.

1.3 Thesis Scope and Research Questions

The work in this thesis investigates the following issues:

- Firstly, this research focuses on modelling approaches, which have become necessary for CAAs to deliver many goals; it also outlines the life-cycle development of CAAs, particularly with regard to obtaining contexts at any time. In addition, it considers requirements engineering, which is an important field for context aware services that are concerned with CAAs' components, to propose several services for users. User requirements call for awareness of environmental objects through CAAs' services, which include objects in the world and achieve the user's needs by representing specific requirements.
- Secondly, this research studies CAAs', investigating a new class of CAAs and their requirements, which need to focus on the differences of functional requirements between traditional systems and CAAs, and also on how to cater for context awareness requirements and carry different contexts according to the user's location and other objects, and express related situations and their changes. In addition, CAAs can interact with the surrounding environment using several context sources around the user in order to be self-adapting.
- Thirdly, this research investigates the limitations of traditional UML diagrams, namely the use case diagram, activity diagram and class diagram, in the field of systems modelling and design, and examines how they can be extended as a new approach. These diagrams are managed by the OMG. UML is used to visualize, modify and design any ISs, including special elements and notations such as use cases, actors, activities, classes and others with special connections between them. Moreover, this thesis focuses on the most challenging issues for CAAs and how to evaluate CAAs using other fields to reduce many challenges, such as complexity in the design and modelling of CAAs.

This approach is extended to encompass CA-UML including the dynamic modelling of CAAs by use case diagrams and activity diagrams as well as structure modelling of CAAs by class diagrams.

CHAPTER 1. INTRODUCTION

In light of the above arguments for the scope of this thesis, this research sets out to answer the following main question:

How can UML extensions be used to model the functionalities, behaviour, and structure of CAAs?

In order to answer the above question, the following sub-questions must first be answered:

- Can the existing use case diagram notations be extended to model the context awareness requirements of CAAs?

If the answer is YES, this poses a new question:

- Can the existing activity diagram notations be extended to depict the dynamic behaviour of CAAs?

If the answer is YES, this poses a new question:

- Can the existing class diagram elements be extended to design the static structure of CAAs?

If the answer is YES, finally:

- Are the proposed extensions of the context-aware use case diagram, context-aware activity diagram and context-aware class diagram practically applicable to real-life case studies?

1.4 Research Methodology

The research methodology is the constructive method which is used in this thesis: this is a customary scientific research technique. This research supports UML diagrams and their tools for various applications to answer the thesis questions as clearly and efficiently as possible. The novelty of this thesis is created using an inventive architecture, modelling, prototype or design with the very high level of expertise that is required in the field of smart systems research [112]. Consequently, the suggested approach of CA-UML diagrams is composed of five work packages, starting with a review of the state of the art and then providing the proposed extension to the use case diagram to model CAAs' functionalities; the third work package presents the proposed extension to the activity diagram to model the behaviour of CAAs; the fourth provides the proposed extension to the class diagram to model the structure of CAAs; and the last work package deals with comprehensive real-world case studies, which are used to emphasize the proposed extensions discussed in work packages 2, 3 and 4.

The work in this research investigates a set of issues in five packages:

1. **Work package one:** Research background to discuss related works and review the literature on the existing approaches in the field of modelling of CAA by UML. This thesis studies the limitations of existing approaches and CA-UML approach to specify the necessary extensions in the following chapters, which enhance CAA modelling. It also searches for related approaches and compares them with CA-UML using scientific journals, books and other publications.

2. **Work package two:** CAA functionalities modelling using the context-aware use case diagram notion. This notion depicts CAA functionalities and scope but is still limited by traditional notations of the use case diagram and needs an extension approach with more concepts suggesting new notations to express the functional requirements and context awareness requirements of CAA and their relationships. Furthermore, this chapter outlines CAA functionalities using practical examples to express the theoretical studies of the CA-UML approach. In addition, CAA functionalities are expressed using clear

CHAPTER 1. INTRODUCTION

models to illustrate the user needs and CAA requirements in a way that is clear and accurate for CAA designers. It also specifies context-awareness requirements to provide developers with a clear understanding of the user requirements.

3. **Work package three:** CAA behaviour modelling using the notion of the context-aware activity diagram. CAA behaviour modelling using activity diagrams is still poor and the limited approaches need to focus on the behaviour activities for CAA and context source and on the relationship between them. In addition, CAAs' behaviour modelling needs a new swimlane and meta-swimlanes which support the interactions between users, context sources and CAAs. It is important to suggest new notations within the CAA swimlane to model adaptation actions and their constraints, and also to visualize how CAAs' behaviour provides the developers with a clear understanding of the adaptation activities and how contexts affect the final actions of CAAs.

4. **Work package four:** CAA structure modelling using the context-aware class diagram notion. The structure of CAA will be designed using a class diagram in preparation for coding, which will demonstrate all details of the CAA attributes and the functions needed to implement CAA. Traditional class diagrams also provide structural aspects for CAA objects as classes, which are able to show the main services of CAA in clear operations, attributes, objects and so on. However, class diagram elements are still unable to present the context structure, which is needed to fulfil the class needs. A new shape is suggested for this notion, describing the context's properties and functions, and a new relationship of <<Utilization>> is necessary, linking the classes of CAAs and contexts. The design of context objects also provides the developers with a clear understanding of the CAAs' structure before the construction level.

5. **Work package five:** Real world case studies of navigation system and weather forecast system using the thesis approach of CA-UML diagrams. In this package, the thesis uses CA-UML diagrams to design a comprehensive modelling of the navigation system and weather forecast system functionalities, behaviour and structure using the notions of the context-aware use case diagram, context-aware activity diagram and context-aware class diagram and their existing and new notations.

1.5 Thesis Contribution

This thesis investigates the requirements engineering and modelling of context aware applications and their weaknesses in adapting to context information changes. It additionally discusses the modelling limitations of UML and investigates the modelling of CAA by UML, with new concepts creating new notations to depict the behaviour and structure of CAA and the reasons for their failure.

New concepts and notions are presented for CA-UML diagrams based on UML to enable the modelling of the functionalities, behaviour and structure of context aware applications, with a focus on context aspects, and their connection to CAA is illustrated. In addition, it is important to investigate different approaches to CAAs related to requirements engineering and system analysis to identify the modelling shortfalls of UML for CAA. Requirements engineering is necessary for any software, especially for CAAs, to outline the requirements specification and validation. CA-UML diagrams propose a new modelling solution for CAA functionalities, structure and behaviour which is able to respond to the environment and to changes in context information and enhance the CAA analysis of contextual information.

The biggest challenge in the CAA environment is the ability to continuously capture changes in context information, which can cause failures and unavailability responses. Although various UML modelling approaches for CAAs have been investigated, they are still limited in terms of CAAs' functionalities, structure and behaviour. More importantly, this thesis extends UML diagrams for context-aware application modelling: these extended diagrams are called CA-UML diagrams. Three notions are proposed, namely context aware use-case diagrams, context aware activity diagrams and context aware class diagrams.

The contributions of this research may be outlined as follows:

- Extending the existing notations of the use case diagram is necessary to model each functional requirement and context-awareness requirement of CAAs and specify their scope. Moreover, CAAs' functionalities need to be illustrated

CHAPTER 1. INTRODUCTION

using new notations of contexts and context sources which express the requirements of CAA. The notion of the context aware use-case diagram provides three concepts: the first is the concept of the adjustable use case diagram, which adjusts existing notations to specify the functional requirements of CAAs modelling; while the second is the use context diagram, which creates new notations to specify context-awareness requirements of CAAs' modelling; the third concept is the context aware usage scenario template, which documents functional requirements and context-awareness requirements for context-aware applications. However, the notations for both concepts are merged for this notion to express all functional requirements and context-awareness requirements of CAAs and link them via a new <<utilize>> relationship. This supports the CAA modelling needed to deliver specific requirements; it also helps to provide the specification requirements of CAAs.

- Extending UML notations for the activity diagram to describe the behaviour of each function for CAA via adjustable levels to depict the real behaviour of CAAs in an activity diagram with new notations within CAA and context source swimlanes and their meta-swimlanes as controllers of CAA activities. In other words, this research investigates an approach involving a context aware activity diagram expressing the CAA behaviour at two levels of standard modelling: the first level of modelling, for general modelling of CAA activities, called high-level interactions, provides a new swim-lane for context source to specify which context source is needed for CAAs; the second level of modelling, for descriptive modelling of Sub-CAAs, users and context sources, is called low-level interactions and provides different Sub-CSs by many users which can explain the interactions between different CAAs or context sources.

In addition, this notion outlines two concepts: the first is the adjustable activity diagram, which adjusts existing notations to describe the behaviour activities of CAA and context source; the second is the context activity diagram, which creates new notations to outline context objects, adaptation actions and their constraints for CAAs. However, both concepts' notations are merged for this notion to demonstrate all dynamic behaviour of CAAs' modelling.

CHAPTER 1. INTRODUCTION

- The Class diagram notations are extended to design structural components of CAA and prepare the functional requirements and context-awareness requirements of CAA for the implementation stage, which demonstrates the extended notions of the context aware use-case diagram and context aware activity diagram to specify their structure.

This helps to clarify the structure modelling of CAA and its components, such as context structure. In addition, the context aware class diagram notion provides two concepts: the first is the adjustable class diagram, which adjusts existing elements to design the structure components of CAAs; while the second is the context class diagram, which creates a new shape to design the structure partitions of contexts and their properties and functions. However, both concepts' elements are merged for this notion to design all component structures of CAAs' modelling and link them via a new <<utilization>> relationship.

1.6 Measures of Success

Measures of success in this thesis are reported as follows:

- The research questions have been answered.
- The new approach for CA-UML diagrams to model the static and dynamic aspects of CAA shows better modelling than existing approaches.
- CA-UML diagrams are presented that express how to depict the functionalities, behaviour and structure of CAA.
- A set of concepts has been introduced for CA-UML diagrams.
- The advantages of new notations for CA-UML diagrams and how to use them have been demonstrated through real case studies and practical examples.

1.7 CA-UML Diagrams

The extension approach developed in this thesis, named CA-UML, includes a standard modelling for use case diagrams, two levels of standard modelling for activity diagrams and a standard modelling for class diagrams. CA-UML diagrams represent an extension approach of UML diagrams for CAA modelling as well as defined extension notations as controlling notions of CAA functionalities, behaviour and structure, including specific conditions for different modelling purposes. CA-UML represents the CAA requirements and environment factors which affect CAA behaviour and structure.

However, while UML is a common modelling language including a set of useful notations expressing system components, it requires further investigation to fulfil the requirements of traditional systems and intelligent systems. In other words, the CA-UML approach is emerging as a set of suggested new concepts and notations along with the powerful modelling language of existing UML diagrams and their notations to fulfil CAAs' requirements and needs.

In addition, CA-UML diagrams have the ability to represent contextual situations for CAAs, which include the common diagrams for achieving functional requirements and context-awareness requirements which describe contexts and context sources as well as suggesting behaviour notations and their constraints to control all CAAs' activities. Furthermore, UML diagrams can be extended to model CAAs' structure and represent all aspects of context-awareness requirements and changes depending on the user environment. Simulating the real-time behaviour of CAA is the most challenging aspect, especially when the output is affected by changes to CAA behaviour and the user's intentions, which are always changing and are detected by different context sources.

A further challenge when designing CAA models is the fact that their modelling tools are still limited in terms of specifying accurate requirements and require adequate modelling and simulation language (such as UML) as well as professional modelling tools (such as Rational Rose or MS Visio).

CHAPTER 1. INTRODUCTION

This research investigates the simulation of CAA using the new approach of CA-UML, which extends the basic UML diagrams to depict the user requirements and the scope of CAA. The first step of extending UML for CAAs modelling is to specify the main components of CAAs and to investigate the limitations of UML in relation to CAA requirements and needs.

UML diagrams do not include any notation for context source and their information (context information), and this is enough reason to take an extension approach to UML with new notations specified for CAAs' requirements. In addition, the purpose of CA-UML is to depict CAAs' requirements (using a standard modelling of the use case diagram), CAAs' behaviour (using two levels of standard modelling for the activity diagram) and CAAs' structure (using a standard modelling of the class diagram).

In this research, context is defined as any knowledge that the CAA needs to capture around the user environment which is extracted from the physical environment and transformed into virtual information available for users. Moreover, it can be any information describing the situation of objects and entities, such as location, person, time, temperature and so on.

Context information which refers to context is modelled using a dotted notation in the CA-UML approach to support CAAs' modelling, simulation and documentation. Depicting CAAs' requirements for context and relevant context source properties enhances the CAA architecture when using special notations expressing CAA behaviour and structure. In addition, the use of context information notation in CA-UML means that context can be sensed by context source. Context can be categorized into two types as follows: atomic context information and composite context information.

Chapter 3, 4 and 5 of this thesis describe the extended notions of CA-UML diagrams in more detail and provide examples, while Chapters 6 and 7 investigate CA-UML modelling for real world case studies of navigation system and weather forecast system.

1.8 Thesis Outline

This thesis is organized into eight chapters with the following objectives:

The introduction is the first chapter, including the thesis research background, methodology, contribution and motivation.

Chapter 2 – Literature Review: This chapter provides an overview of the importance of requirements engineering and CAA and reviews the previous works on existing UML approaches to model CAAs and their limitations as well as the advantages of simulation of the CAAs' behaviour and structure.

Chapter 3 – An Extension of the Use Case diagram to Model Context Aware Applications: This chapter presents the first part of the CA-UML diagrams, which specifies the notion of context aware use-case diagrams providing functional models of CAAs. This notion merges the existing notations of the use case diagram and the new notations of the use context diagram to describe CAA functionalities. CAA functionalities are divided into functional requirements, which are represented by the adjustable concept of the use case diagram, and context-awareness requirements, which are represented by the new concept of the use context diagram. Both are able to present context information and context source.

Chapter 4 – An Extension of the Activity diagram to Model the Behaviour of Context Aware Applications: This chapter provides the second part of the CA-UML diagrams, which specifies the notion of the context aware activity diagram, describing the behaviour models of CAAs. This notion merges the existing notations of the activity diagram and the new notations of the context activity diagram to demonstrate the dynamic behaviour of CAAs and suggests a two-level standard modelling which controls the specific activities for each component as a swimlane and meta-swimlanes. Although the existing activity diagram includes two swimlanes for users and the system only, CAAs need an extra swimlane to include context source as well as to depict the internal activities for each component in order to express CAAs' behaviour thoroughly.

CHAPTER 1. INTRODUCTION

Chapter 5 – An Extension of the Class Diagram to Model the Structure of Context Aware Applications: This chapter outlines the third part of the CA-UML diagrams, which specifies the notion of the context aware class diagram designing the structure models of CAAs. This notion merges the existing elements of the class diagram with new elements of the context class diagram to design the component structure of CAAs. However, a new shape depicting contexts' properties and functions will improve the awareness of CAA components as well as using the same method of inheritance between different classes. In addition, a new relationship is suggested to link classes and contexts to help to retrieve contexts' values for classes of CAAs.

Chapter 6 – Real-life Case Study of Context-Aware Navigation System using CA-UML diagrams: This chapter investigates the technical aspects and theoretical considerations of this thesis and outlines the pragmatics of the CA-UML approach, which are evaluated using a realistic case study of a navigation system expressing the practical usage of a set of notions, namely the context-aware use case diagram, context-aware activity diagram and context-aware class diagram, to depict CAAs' functionalities, behaviour and structure.

Chapter 7 – Real-life Case Study of Context-Aware Weather Forecast System using CA-UML diagrams: This chapter describes the technical aspects and theoretical considerations of this research, which are evaluated using a realistic case study of a weather forecast system expressing the practical usage of a set of notions, namely the context-aware use case diagram, context-aware activity diagram and context-aware class diagram, to depict CAA functionalities, behaviour and structure.

Chapter 8 – Conclusion and Future Work: This chapter summarises the thesis's findings and presents a conclusion, and also suggestions for future work to extend CA-UML diagrams to include all UML diagrams that are able to describe the dynamic and static aspects of CAA.

2. Literature Review

Objectives:

- Describe the requirements engineering concept and the differences between requirements engineering approaches, types, processes and categories;
- Explain the importance of CAA modelling and simulation;
- Investigate approaches for the conversion of physical to virtual environments;
- Introduce Context Information, Context Sources and Context Aware Systems;
- Critically review related work on UML and CAA.

2.1 Introduction

This chapter provides an overview of RE, UML and CAA and their important aspects by grouping them together. This chapter also presents simulations of CAA behaviour and structure. System modelling uses technical languages to depict system structure and behaviour and to express systems' requirements based on specific conditions and rules such as key value models, markup schema models, graphical models, object-oriented models, logic-based models and ontology-based models [49, 68, 69].

In addition, this chapter provides brief definitions and descriptions of CAAs and discusses the SDLC of traditional systems and how to evaluate requirements engineering and modelling for CAAs and the relationship between them. CAAs represent an interesting and challenging field in different subjects. However, requirements engineering of CAAs has many challenges: the main problems with CAAs involve capturing contexts, connection attributes, context source sensing and the user environment, which are always changing [66, 70].

CHAPTER 2. LITERATURE REVIEW

CAAs need different types of contexts, which refer to any information and knowledge that expresses a situation relating to the interaction between users, systems and the environment. Context source is also the provider of contexts that are used to refer to physical or virtual sensors, while context information refers to the sensed data extracted from user environments to be carried into supposed systems [41, 57, 63].

Moreover, it is important to investigate the uses of SDLC for traditional and smart systems; the beginning stages of SDLC are requirements gathering and modelling, which are necessary for CAA and which follow other steps of analysis, construction, testing and implementation for CAAs using UML notations to capture the contexts depending on CAA requirements.

Finally, this chapter investigates the background of UML and CAA modelling, outlines the differences between existing approaches used to model CAAs, compares these approaches with CA-UML and defines the limitations and weaknesses of each approach [10, 31].

2.2 Requirement Engineering and Modelling

SE is becoming increasingly popular, as it offers a useful way to focus on system quality and performance. The first stage in SDLC is requirements engineering and modelling the defining system tasks, as well as each step and the functionality of any system, and identifying system requirements earlier to fulfil the user's needs [19].

Why is system modelling an important stage of the Software Development Life-Cycle?

The modelling and simulation stage for such ISs is a way to express the system planning and objectives and to specify the system tasks, which provide more details about the system before the construction stage (coding stage) to save time and effort; modelling, the design and analysis phase is the most important stage of any system to resolve system weaknesses early on. Furthermore, SDLC is important for system quality and performance, which requires a creative team to coordinate with each other and specify

CHAPTER 2. LITERATURE REVIEW

the team tasks, such as setting out the system requirements, system design, system coding and building, system testing, implementation and maintenance. SDLC provides a set of system methodologies to enhance all software stages to increase the system quality, such as the Waterfall Model, the Spiral Model, the Concurrent Development Model, the Agile Model, the Incremental Model and so on [54, 56].

There are many modelling languages designed for systems and applications, such as Bigraphical Modelling Language, Unified Modelling Language, Specific Modelling Language, Algebraic Modelling Language, Virtual Reality Modelling Language, Object Modelling Language and so on. In addition, current studies and investigations are developing special modelling approaches to depict the environmental factors which affect systems' structure and behaviour [34, 73, 104].

Moreover, transferring physical objects to virtual form and carrying them to many users will help to fulfil the users' requirements and needs. The main role of the software is to specify the system's functional and non-functional requirements as well as system functionalities, tasks, constraints, services and other aspects executed by the system and their development processes [40, 53]. Software process models provide different kinds of model of the software development process. The common models used for different purposes are as follows:

- The Waterfall Model involves sequential processes working as a waterfall which consists of various phases from the design phase to the maintenance phase.
- Evolutionary Process Models provide more methodologies for Software process models, such as the Prototype Model and the Spiral Model.
- The Prototype Model is the most popular and widely used model, as it offers a preferable way to create abstract system services as initial demonstrations for many users as the first step to fulfil user needs but without system details. This is a helpful way for software developers to gather customers' feedback early.
- The Spiral model is a software development process that combines the prototype concept, the top-down concept and the bottom-up concept.

CHAPTER 2. LITERATURE REVIEW

Most of the existing models of software development processes are suitable and defined to address the traditional system problems and development, but are still limited for CAA development. This thesis investigates the first stage of requirements engineering for CAA, which addresses CAA modelling needs using UML as a modelling concept and Microsoft Visio as a modelling tool [118].

Furthermore, the Bigraphs model considers solutions for Ubicomp and enhances CAA requirements, and provides a platform for independent agents using ambient mobile calculus. In addition, the Bigraphs model represents special privacy functions. Milner conducted a set of studies to investigate the importance of privacy in ubiquitous surrounding interactions using calculi processes as well as different programming languages and simulation [71, 72, 105].

Table 2.1: Examples of project failure

Project Name	Failure Reason	More Information
London ambulance service dispatch system	Poor requirements analysis	Closed down in 1992 after two days' operation
Wessex regional information systems plan	No clear definition of system scope	Abandoned in 1990 after spending 43 million pounds
London stock exchange TAURUS	Failures to reconcile conflicting requirements	Cancelled in 1993 after spending 75 million pounds
Performing rights society	Poor requirements engineering	Abandoned in 1992 after spending 11 million pounds

CHAPTER 2. LITERATURE REVIEW

The modelling stage in SDLC represents the system specifications which express the tasks and services as models including all software aspects during system development. The modelling phase also translates system requirements, owner ideas and user needs to technical plans.

In addition, requirements engineering aspects resolve the main causes of project failure early (see examples of project failure in Table 2.1). Furthermore, drawing information systems helps to clarify the future needs and requirements that are specified by different modelling concepts, including system architecture, high level design and system prototypes, which support communication with stakeholders in the design phase.

2.2.1 Requirements Gathering and Analysis

Requirements engineering and modelling provide different approaches and methodologies for increasing system performance. The first stage of the SDLC is requirements gathering and analysis (Requirement Engineering). Requirements gathering and analysis focuses on system planning to present different models for designers, and also validates system requirements before the coding phase [38, 39].

System requirements are collected in different ways, including interviewing customers to find out their requirements, distributing questionnaires to specify the user needs and prototyping to gather users' feedback and comments. System requirements may be classified into three types: user requirements, system requirements and system design specifications. Table 2.2, below, shows the differences between requirement types and identifies the requirements reader for each type.

CHAPTER 2. LITERATURE REVIEW

Table 2.2: Requirement types

Requirement Type	Description	Reader
User requirements	Simple sentences explain user needs executed by information system and include conditions	<ul style="list-style-type: none">- System Architect- Contractor Manager- System End-user
System requirements	Structure document expressing the system services, specified as a contract between the system user and the system provider	<ul style="list-style-type: none">- Software Developer- System Architect- Client Engineer- System End-user
System design specifications	Technical documentation describing the specifications of system components	<ul style="list-style-type: none">- Client Engineer- System Architect- Software Developer

Requirements methods are investigating a set of modelling and analysis approaches that attempt to capture all requirements for such systems and identify the system problems from the user community and suggest suitable solutions. Requirements gathering provides different methodologies such as Fact-Finding Techniques, Model-Driven Analysis and Accelerated Systems Analysis as follows.

CHAPTER 2. LITERATURE REVIEW

- **Fact-Finding Techniques**

Fact-finding is the collection process which gathers the information needed for a system and finds solutions, opportunities and chances to address system problems and challenges. In addition, fact-finding techniques provide solutions for system documentation, system databases, system reports and system models. This approach focuses on the system requirements depending on users' needs, and on the relationship between users, administrators and the technical team. Fact-finding techniques represent a prototyping model to allow users to verify system requirements as well as using survey approaches such as interviews and questionnaires to collect the required information through the user community to specify which system actions users need [79].

- **Model-Driven Analysis**

Model-driven analysis is an approach designed by the OMG which uses UML diagrams to model static and dynamic systems and to specify business problems and their solutions, user requirements, and system scope. Furthermore, it involves a set of approaches to analyse and depict system structure, such as OOA [85, 91].

Model-Driven approaches are always enhanced by the use of professional software tools such as Microsoft Visio, Rational Rose, System Architect and Visible Analyst.

Some analysts provide prototyping examples for the users or software owner, which require feedback before the coding and implementation phases. System analysts prefer to use CASE tools.

- **Accelerated Systems Analysis**

The accelerated systems analysis approach evaluates existing prototypes to identify the user requirements and upgrade incomplete system prototypes. Accelerated systems analysis provides development models for system architecture, simplifies complex system prototypes and resolves misunderstandings between system end-users, system analysts, system developers and system programmers.

CHAPTER 2. LITERATURE REVIEW

Furthermore, accelerated systems analysis designs logical models which depict the user requirements in technical models and execute those requirements to system codes for system implementation and configuration.

2.2.2 Requirements Engineering Processes

RE describes exactly what the system should do and the results of failure in terms of time and budget in the following stages of the system, which might involve complete implementation failure, necessitating that the project be re-started from the beginning.

Many researchers are interested in requirements processes at the system analysis and design stages (as shown in Figure 2.1). User requirements are the key to analysis and building of any system or application and should focus on this part before analysis and construction of the system or application to save time.

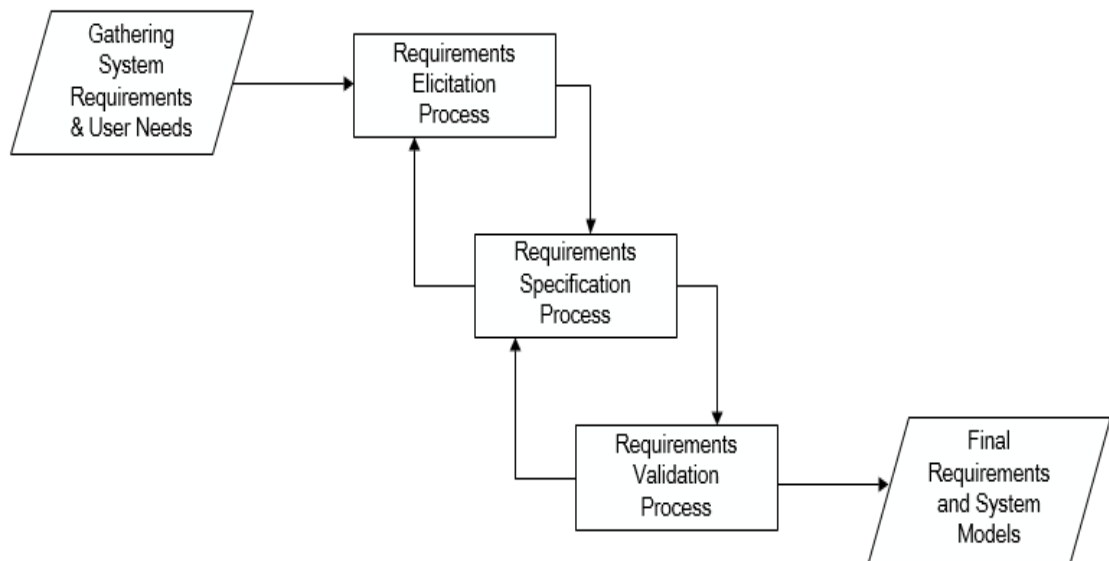


Figure 2.1: Iterative processes of requirements engineering

The processes involved in requirements engineering consist of three main activities: the requirements elicitation process, the requirements specification process and the requirements validation process.

CHAPTER 2. LITERATURE REVIEW

- Requirements Elicitation Process

The requirement elicitation process is the first process of requirement gathering which collects all information needed by discussing the stakeholders' requirements and needs, understanding the customer's problem and identifying the system services and solutions in more detail. The requirements elicitation process involves collecting all information from the users and describing the user needs as well as specifying what the customer's requirements are.

In addition, the requirement elicitation process is concerned with gathering and identifying the user requirements. Thorough requirements elicitation and collection support the requirements specification process. Furthermore, the requirement elicitation Process investigates the problem in detail and suggests solutions, as well as helping to analyse the business requirements and study the reasons for the problem by using different data collection techniques such as interviews, scenarios, questionnaires, and prototyping.

- Requirements Specification Process

The requirements specification process is the most important and interesting process at the requirement analysis and design level. This process also specifies and outlines the requirements as a last step before system modelling and coding, which is important to ensure completeness.

The requirements specification process is a commonly confused process during the requirements engineering stage which needs validation and testing to ensure that system requirements are carefully distinguished and described and to identify the necessary system functionalities. Furthermore, the requirements specification process illustrates the important requirements and a good requirements specification produces a good requirements model, which supports system services and quality. Experts recommend a focus on a requirements specification process which clearly describes the requirements to system developers.

CHAPTER 2. LITERATURE REVIEW

- Requirements Validation Process

The requirements validation process is the process that tests the specification of system requirements and verifies that the product will meet the system requirements. In addition, the requirements validation process ensures that none of the requirements specified in the requirements specification process are missing and checks the requirements of system functions and services. Furthermore, the requirements validation process checks the user requirements for completeness, with a focus on accuracy and validation to ensure that the system is easy to maintain in the early stages to get accurate results.

The requirements validation process confirms the system results and actions for users, observes and reviews the system requirements, checks that all requirements are well defined and compares them with system output at an early stage. This can improve system quality as well as providing planning for system requirements verification, system testing, system implementation, system maintenance and system auditing.

2.2.3 Requirements Categories

In software engineering, requirements may be categorized into functional requirements and non-functional requirements as follows:

- Functional Requirements

Functional requirements outline the requirements of static and dynamic systems and define their components as well as expressing the system, functions, and objectives and preparing for system architecture. Hence functional requirements define the system inputs, processes, output, structure and behaviour. In addition, functional requirements identify the system services as well as how the system interacts and responds in different situations and what actions should be taken with conditions.

CHAPTER 2. LITERATURE REVIEW

In other words, functional requirements specify the scope of the system and its functions, which means that the system should execute functional requirements, and also outlines what the system must do to fulfil the system requirements and implement these functions, which are mandatory requirements to run system services.

Furthermore, functional requirements should define and describe the system requirements and the technical knowledge in such a way that they can be clearly understood, such as input data (system parameters), system operations, system workflows, system outputs, system reports and system documents.

Functional requirements outline the input data, processes and system outputs. IPO is the most common model used to describe the system processes and convert data to information.

The main aspects of functional requirements can be described as follows:

- **Input Requirements**

Processing the data by collecting different inputs for specific purposes which are required to create results and information. Furthermore, the system inputs for a set of requirements, such as user requirements, system requirements, conditions and constraints.

- **Interface Requirements**

The system interface is the interaction component between the user, software, hardware and their communications. In addition, the interface should be concerned with the behaviour of interaction and should suggest different options.

- **Operation Requirements**

System functions are the processing stage of input data and specify how the system executes entered data. System operations possess a set of processes for the data which run the requirements and set their conditions.

CHAPTER 2. LITERATURE REVIEW

- **Output Requirements**

The system produces final results depending on specific inputs and processes. The changes of output requirements will impact on IPO activities, which change the process rules and input data.

- **Non-functional Requirements**

Non-functional requirements provide system quality objectives that are related to functional requirements such as system reliability, system usability, system performance and system security. Software developers are not usually interested in security requirements at the design, analysis and construction stages of the system. In addition, it is better to specify sensitive requirements at the early stage of software building, especially at the modelling and analysis stage, to improve awareness of future risks. System security may be identified as non-functional requirements for information systems, and various specialists and researchers agree with this definition. System security is a form of information technology that serves to protect system information against any threat, attack or unauthorized access [88].

In addition, information security is the science and policies involved in protecting information from risks or attacks, while system security provides the necessary tools to protect system information from internal or external risks. Furthermore, security policies protect the privacy of user information, and their most important task is to monitor system information and ensure that system content has not been modified by unauthorized users and that it can interact only with information that is available to all authorized users [95, 97, 98].

Many researchers and specialists are interesting in security requirements, and the recommendation is to specify the security requirements at this stage.

2.3 System Modelling, Design and Simulation

System designing, modelling and simulation are important steps before the implementation of any system (as shown in Figure 2.2). System configuration is the last step, and such a system will cost a huge amount and waste considerable time in the case of system failure or the detection of many problems and mistakes, especially in complex systems [20].

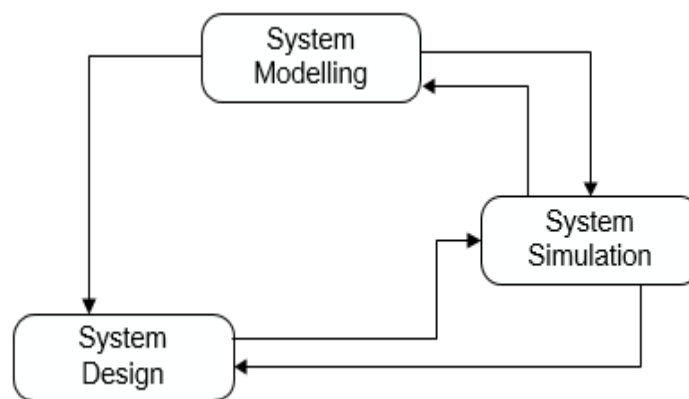


Figure 2.2: Iterative processes of system modelling, design and simulation

Simulating different scenarios to produce models for CAA requirements will increase CAA quality and performance, and also reduce various challenges when using CAA services, such as time and cost. An object is any physical thing that can be touched, and sensor devices for different uses usually capture specific context information such as location, vehicles, buildings, human beings, devices and so on.

Transformation is the main aim of CAAs. The main challenge for CAAs is to transfer our real life or the physical environment into a virtual environment, and in recent decades there have been thousands of studies focusing on CAAs to successfully achieve this goal to interact with our lives in ways that are easier to manage, leading to many opportunities to succeed in tasks by using available environment monitoring. Recently, CAAs have been making major technological advances with smart devices that help users to achieve many business processes by using integrated applications [74, 77].

2.3.1 CAA Behaviour Simulations

CAAs simulation using new approaches to help CAA designers is necessary to find out about the behaviour of CAAs and their integration with all software and hardware. Simulating the real-time behaviour of CAA is the most challenging part, especially when producing the output is affected by changes of CAA behaviour and the user intentions, which are constantly changing and are detected by different CSs [78].

Understanding system behaviour is a common challenge for many researchers, as well as system behaviour modelling and simulation, especially for complex systems such as CAA. However, CAA modelling is still approached using different methodologies and there is a need to investigate several aspects, such as CAA run-time behaviour and networks [109, 110]. However, CAA behaviour simulation is affected by different factors such as user environments and resources; also, the user's needs may be changed through the run time of CAA, leading to contexts changes (as shown in Figure 2.3).

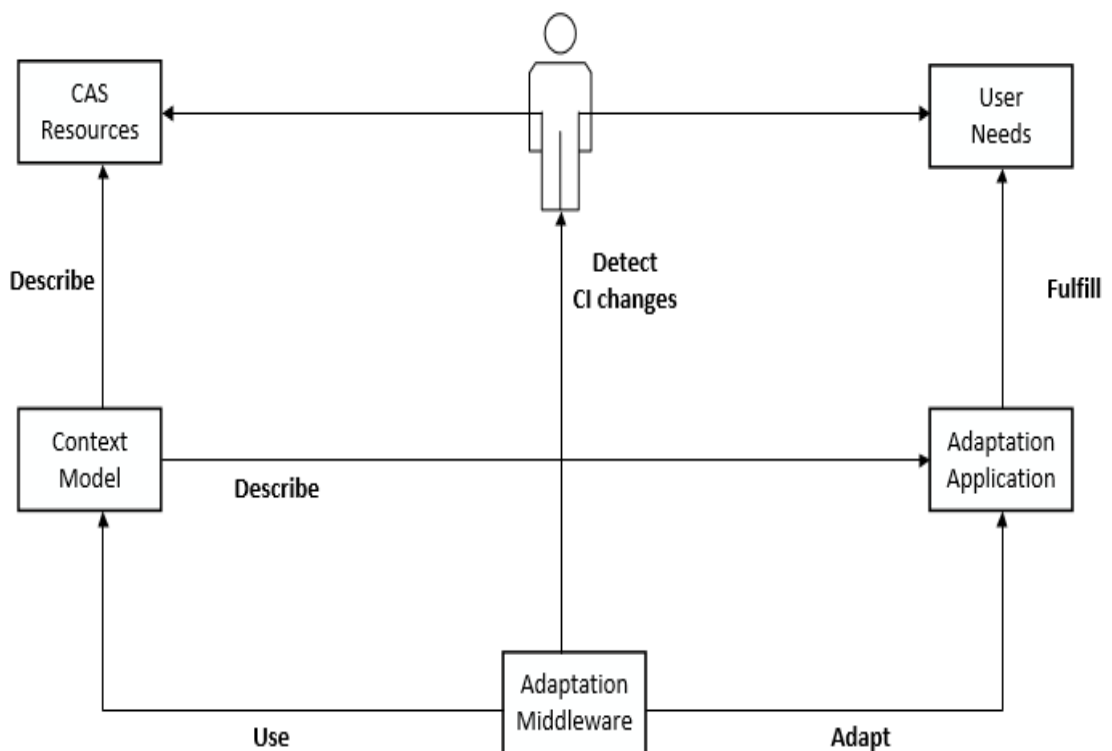


Figure 2.3: CAA Behaviour Simulation

CHAPTER 2. LITERATURE REVIEW

Depictions of the behaviour of CAA, context source and the user environment are still poor and need more investigation to study the conversion of physical to virtual entities that are needed to support CAA functionalities (as shown in Figure 2.3), and also to express the interactions between the user, CAA and context source [7, 18]. Some CAA consists of many subsystems, as well as other elements of software, hardware and electronics. CAA needs to realize all parts to provide the user with specific information at any time and place.

2.3.2 CAA Structure Simulations

Simulation of CAA will define which part is important for CAA, specify the purpose of any interaction with an external part and identify different options for CAA 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 [75]. CAA structure simulation is necessary to express the different parts connected with CAA and to depict parts in relation to each other, representing a set of objects or components (as shown in Figure 2.4).

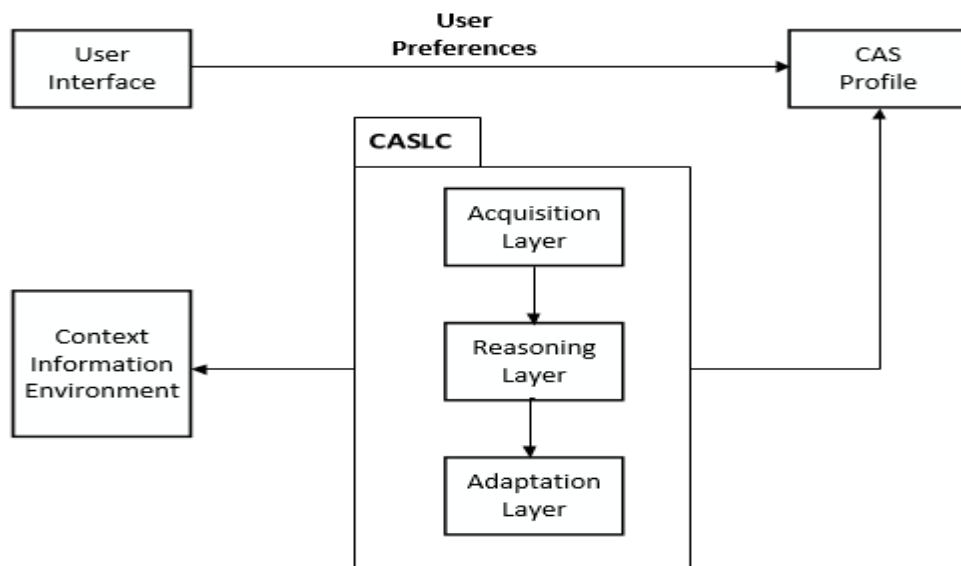


Figure 2.4: CAA Structure Simulation

2.4 UML

UML is a diagramming language used for modelling and designing maps for systems or applications; it is called unified because it standardizes several slightly disparate elements. UML is a common modelling language that was created by the OMG in 1997. There are many types of UML diagram, such as use case diagrams, activity diagrams, sequence diagrams, class diagrams and so on. It is recommended to focus on the interaction between UML diagrams' connections of use cases, actors, attributes, classes, objects and methods [17, 60, 107].

It is important to integrate them with each other to make comprehensive maps of any system in preparation for errorless construction. However, UML diagrams have the ability to model all systems' structure components and dynamic behaviour.

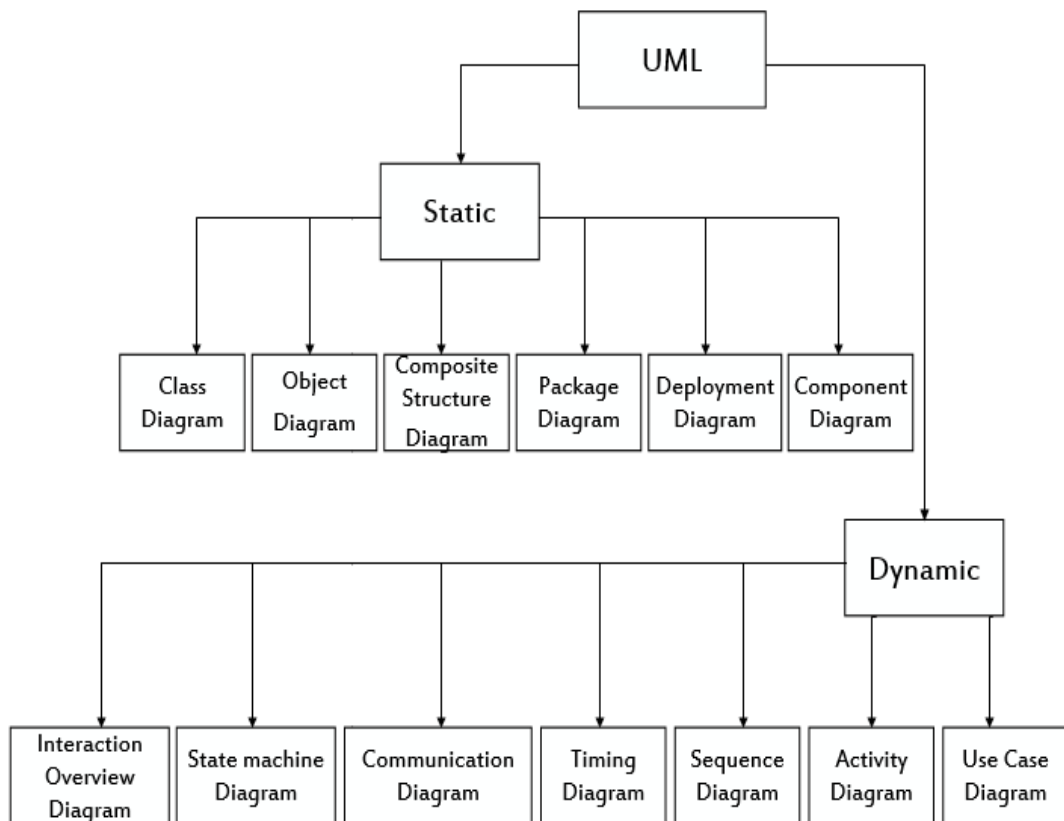


Figure 2.5: UML Categories

CHAPTER 2. LITERATURE REVIEW

Timing is important at the requirements analysis and modelling stage. For example, UML sequence diagrams can specify the system interactions as a sequence of messages which interact with each other as objects and entities.

In Figure 2.5, we can observe the categorization of UML diagrams into structure diagrams and behaviour diagrams. Structure diagrams (static diagrams) model the system's static components in different structure modelling for the system and show the system parts and their interactions as objects, classes, interfaces and so on. Behaviour diagrams (dynamic diagrams) model the system's dynamic components and their aspects, which express different actions, as well as the interactions within the system components or external users.

UML diagrams possess the professional solutions for IT modelling and system requirements to design different types of software applications and their components, which include structure and behaviour design. UML diagrams help to understand system needs and provide a set of models expressing the system abstract in high level modelling, and also provide system detail in low level modelling. Furthermore, UML outlines the communications between systems and stakeholders and specifies their requirements as efficient models which explain each system component and its needs (Table 2.3 expresses a comparison between UML diagrams).

In addition, UML diagrams are of interest for many researchers in developing their ability to model and design all types of system, especially intelligent systems, which need more research so that they can be extended for different purposes [61].

Typically, UML 2.0 consists of thirteen static and dynamic types of diagram (as shown in Figure 2.5) and the latest version of UML, 2.4.1, may include more diagrams.

CHAPTER 2. LITERATURE REVIEW

Table 2.3: Comparison of UML diagrams

UML diagram	System structure	System behaviour
Use case diagram	×	√
Activity	×	√
Class diagram	√	×
Component diagram	√	×
State machine diagram	×	√
Sequence diagram	×	√
Deployment diagram	√	×
Communication diagram	×	√
Interaction overview diagram	×	√
Object diagram	√	×

CHAPTER 2. LITERATURE REVIEW

Timing diagram	×	√
Package Diagram	√	×
Composite structure diagram	√	×

2.4.1 Use Case Diagram

Use case is the most popular of the UML diagrams, describing the system functionalities and the interactions with the user via a graphic description in sequence steps. Traditional use case diagrams show a general view of traditional system functionalities and do not provide more details [48]. The use case diagram abstracts the system functions and summarizes the main software components and the relationships between them, such as actors, use cases and external systems (as shown in Figure 2.6).

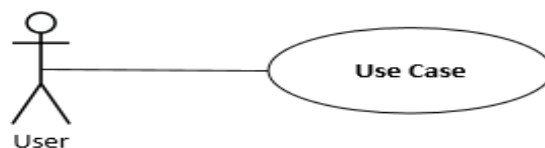


Figure 2.6: Use case diagram notations

In addition, use case diagrams identify the system functionalities as the following notations:

- Actor notation: depicting the stakeholder actions that interact with system functions as well as actors, which can be external systems, organizations and devices.

CHAPTER 2. LITERATURE REVIEW

- Use case notation: expressing the system functionality, which provides user needs and requirements, such as login, send a message or download a document.

Furthermore, the scope and boundary of the use case represent the system's functionalities and the main system components and external parts which interact with each other.

Specifying a set of relationships which define the relationship type as follows:

- Generalization relationship: Actor to Actor relationship
- Dependency relationship: Use Case to Use Case relationship, including two Dependency relationships: <Include relationship> and <Extend relationship>
- Association relationship: Actor and Use Case relationship

Use cases and their scenarios are not the same process and the main difference between them is that the use case expresses the system actors in general but the usage scenario template describes who the actor is and provides details about him, as well as explaining each system's functionality in different steps and stories [51].

2.4.2 Usage Scenario Template

The purpose of the usage scenario template is to provide more details for each use case and outline the important information needed in the design and architecture stages for the system requirements. An example of a usage scenario template (as shown in Table 2.4) involving a use case scenario for Login describes all steps, conditions and details for the specific use case. The usage scenario template can also express a set of use cases which express different scenarios of such a system. Most system developers use a usage scenario template to support the success of requirements definitions for system objectives and increase clarity for system programmers. A usage scenario template is also necessary for documentation and project management [106].

CHAPTER 2. LITERATURE REVIEW

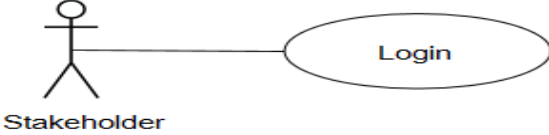
The objective of a usage scenario template is to make collection scenarios for each use case by writing a story of project scope and requirements with deep interactions between the user and the system. The usage scenario template also considers the workflows of the use case. It describes the interactions between the user and the system and manages their actions and steps during each interaction. The use case diagram does not provide more details to abstract the system and the usage scenario template is required to explain each use case in different stories. In addition, the usage scenario template helps to move from the use case diagram to the activity diagram and the sequence diagram. In other words, the use case diagram depicts the system functionalities in diagrams as use cases and the usage scenario template describes each use case in clear words as stories.

An example of a Login use case is shown in Table 2.4. This case is not described well in the use case diagram and needs more explanation to specify how many times the user is able to insert the account login details, such as user-name and password, and also what the system should do if the process goes wrong and what happens when the user inserts numerous incorrect logins. This can happen in different stories and alternative scenarios should be specified to make different actions depending on the interaction between the user and the system.

In addition, the usage scenario template should create documentation for system functionalities: the main elements of the usage scenario template are Use Case Name, ID, Actors, Priority, Pre-Conditions, Post-Conditions, Frequency and so on.

CHAPTER 2. LITERATURE REVIEW

Table 2.4: Usage scenario template

Usage scenario template Elements	Example
Use Case Name	Login
Use Case ID	Give a unique number for each use case for functional requirements and give a unique letter for each use case non-functional requirements
Use Case Diagram	 <p>A UML Use Case Diagram showing a stakeholder connected to a use case. The stakeholder is represented by a stick figure with the label "Stakeholder" below it. A horizontal line connects the stakeholder to an oval use case labeled "Login".</p>
Priority	High
	<ul style="list-style-type: none">- The stakeholder is using the system interface and the first functional requirement is Login- The system displays the login form before the access system functions- The stakeholder inserts a valid user-name and password in login- The system should verify the user's login information

CHAPTER 2. LITERATURE REVIEW

Scenario narrative	<ul style="list-style-type: none">- If the stakeholder inserts incorrect login information- The system displays the login form again- The stakeholder re-inserts a valid user-name and password in login- The system should verify the user login information again- If the stakeholder has inserted incorrect login information- The system exits the interface screen- Else the system allows the authorized stakeholder to access system- The system will display the main system screen- The stakeholder can select an available system function
Actor	Authorized stakeholder
Basic flow	<ul style="list-style-type: none">- The stakeholder inserts the user-name and password- Stakeholder is validated- System displays all available system units
Alternative course	<ul style="list-style-type: none">- The stakeholder is not authorized- The system displays an error message- The system shows the login form again- The stakeholder re-inserts the user-name and password- Stakeholder is validated

CHAPTER 2. LITERATURE REVIEW

	- System displays all available units
Pre-conditions	Stakeholder must be an authorized user and have a valid system account
Post-condition	The stakeholder can access system functionalities to use such services
Frequency	Twice per visit
Future Requirements	Update user-name and password every month
Locate	Use Case Model

2.4.3 Activity Diagram

A UML activity diagram is a behaviour diagram which depicts the system work flows and sequence conditions and provides more detail and actions via activity nodes including object, control and action, as modelled in Figure 2.7. The activity diagram describes the system behaviour as models when the use case diagram and usage scenario template are done.

CHAPTER 2. LITERATURE REVIEW

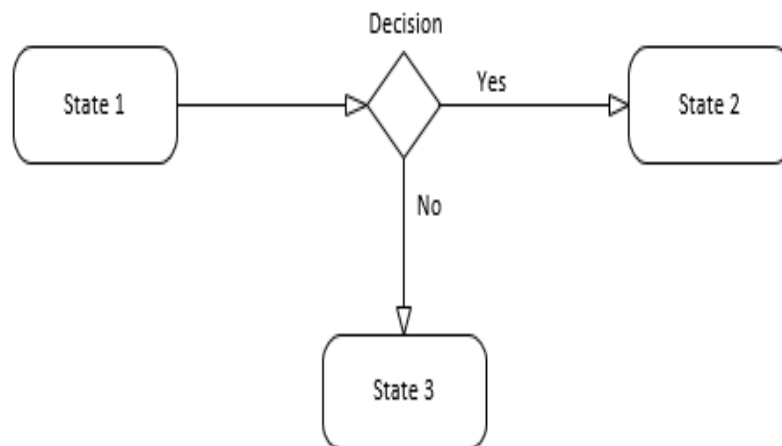


Figure 2.7: Activity diagram notations

The UML activity diagram expresses the sequence work-flows and shows the system behaviour as activities and their actions and conditions. Moreover, the activity diagram is used to demonstrate the interactions between user and system in two swimlanes: one for the user activities and another for the system activities. UML diagrams map the system requirements to diagrams, which are important processes to specify the project objectives and services and need more explanation via different types of modelling to show all system activities [55].

The documentation stage of the usage scenario template is between the use case diagram at high level (system abstract) and the activity diagram at low level (system detail). The usage scenario template should be specified when the use case diagram is complete to explain each use case in the usage scenario template. Based on the usage scenario template of each use case, the Activity model represents the flow of actions, activities or processes which are performed by the system; the activity diagram handles more low level details of system functionalities with a clear illustration of all possible behavioural actions or responses of the system.

CHAPTER 2. LITERATURE REVIEW

2.4.4 Class Diagram

The class diagram is a static model of the system structure which expresses the system components and their relationships and outlines the system classes, attributes (properties), functions (methods), operations and object relationships, as depicted in Figure 2.8.

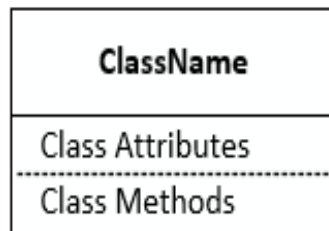


Figure 2.8: Class diagram shape

In addition, class models are necessary to model the user domain elements and the structural parts of the system. The class diagram has one main element, which consists of three parts: the class name, attributes and methods. Each class is given a unique name and class attributes represent the class properties and details as well as the relationship between objects using different types of relationship, such as aggregation or association. Furthermore, the methods part displays the system functionalities, such as ‘send a message by email’ or ‘delete the message’.

Moreover, the class diagram describes the system structure in models to translate the system models to source code, which provides the details needed in the programming stage.

2.4.5 Stereotypes

- Basic UML Stereotypes for traditional system

The basic stereotypes concept enables representation of specific commonalities of classes with each other for clarity and ease of understanding by sub-typing the classes

CHAPTER 2. LITERATURE REVIEW

into sub-classes. For example, the user classes among school system users consist of student, tutor, advisor and manager; in stereotypes, these will be divided into <<user>> student, <<user>> tutor, <<user>> advisor and <<user>> manager.

- Extended UML Stereotypes for CAA

Currently, stereotypes are accepted as a concept for the representation of CAA structure as classes and objects. However, the extended stereotypes for CAA structure modelling help to specify the property types for CAA and their values. CAA stereotypes are advanced class diagrams used for large-scale systems such as CAA and are also widely used for CAA structure modelling. CAA stereotypes approach modelling is an extension of the basic stereotypes.

Using stereotypes becomes necessary for smart applications to illustrate the system partitions (such as the association relationship to link a set of contexts). The concept of stereotypes can also represent the structure components of CAA, helping modellers to specify the association between users and their activities [2].

2.5 Context Aware System

Nowadays, the usage of CAA and their applications and devices is increasing every day, calling for investigation of different services for user requirements and fulfilment of user needs. CAAs are everywhere, and have become part of everyday human life. They are becoming ever more popular, as they can achieve users' needs in different sectors, such as mobile connections, technology, health, education and many more [32, 33, 35, 36].

In addition, Context Aware Systems adapt to many behavioural and structural situations involving users, networks, data, hardware and the application itself. The external environment includes different contexts needed for the user, such as location, time, person and so on, and there is no specific definition of the context. The creation of contexts from different sources supports CAA and their software and hardware

CHAPTER 2. LITERATURE REVIEW

components, which need development to improve their quality and performance, but this is costly [37, 41]. CAA needs new methodologies for making platforms as people, goods and devices become increasingly connected and require interactions to achieve high performance for CAA services, which increase CAA's ability to reflect the best behaviour actions on time without delay or error, representing the real world as it happens and predicting what will happen later on.

Furthermore, CAA manages a set of services by using associated data from different CSs which extract the contexts as parameters and carry it to CAA immediately using intelligent context providers such as GPS, weather services and traffic sensors. In addition, CAA users receive relevant information and recommendations depending on their situation, user preferences and available parameters [57, 63].

To provide greater understanding of the nature of CAAs, the following paragraphs will outline the main constituents of CASLC, which execute the CAA framework as follows:

- Acquisition Layer

This layer collects contexts and senses physical objects as well as monitoring the changes of contexts which may affect CAAs' behaviour. Sensors acting in this layer form a lower layer interacting with the physical environment and their main task is getting contexts and exchanging them with CAAs.

- Reasoning Layer

This layer is the processing stage for contexts and converts physical entities to virtual entities. It is also responsible for interpreting contexts to provide clear information in the Acquisition Layer.

- Adaptation Layer

This layer adapts the services for CAAs to fulfil the user needs and CAAs requirements, and also adapts contexts changes in response to changes in the user's environment. Furthermore, the Adaptation Layer makes decisions and carries out specific actions

CHAPTER 2. LITERATURE REVIEW

depending on CAAs' requirements and user needs. Table 2.5 outlines the common Context Categories as follows:

Table 2.5: Context categories

Context Categories	Identification	Examples
Where	The place of context	City, country and region
When	Define the time of the event	Day, month, season and year
Who	User details	User name, age, gender, address and so on
What	Related objects	Applications, services and systems commands
Why	User intentions	Events and future situations
How	Context processes	Sensors' activities, functions or methods

2.5.1 Context

There are many definitions of context, put forward by different researchers, which attempt to express the exact meaning of context; however, some readers are still confused and need to read numerous studies to understand the exact meaning of this term. It will be helpful to first define the concept of context and then what it means for an application to be context-aware [76].

CHAPTER 2. LITERATURE REVIEW

Dey and Abowd have provided perhaps the clearest definition of context. Some important questions can be asked, such as: "Why does such an application need to capture the context"? To answer this question, it is important to express a set of examples which investigate context types and uses for any system depending on the user needs. 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".

2.5.2 Context Source

Context source is the provider that produces context and carries it to the application and making the transformation processes required for CAAs. The main challenge for CAAs is to transfer our real life or the physical environment into the virtual environment. In recent decades, thousands of studies have focused on CAAs to succeed in this goal, interacting with our lives to make them easier to manage, leading to many opportunities to achieve tasks successfully through the use of the available environment monitoring.

Recently, CAAs have been making major technological leaps, with the development of smart devices that help users to undertake a wide range of business processes through the integration of applications. Context source is still rarely used in current applications and information technologies need all contexts types which upgrade and convert the traditional systems to intelligent systems to make life easier and faster. Context source modelling is poorly investigated by UML diagrams and more research is needed to provide a set of results which resolve the various challenges for CAA modelling. In addition, understanding CAA requirements and user needs will help to specify contexts and how they can be used by clear CAA architectures and provide good modelling for CAA behaviour and structure, expressing the real environmental situations to support CAAs' building and implementation [83, 96].

2.6 Existing UML approaches for requirements modelling of CAA and their limitations

Researchers in the field of intelligent systems, especially in the domain of CAA, have adopted different philosophies about how to correctly simulate information systems using UML approaches. The following paragraphs outline the approaches that are most relevant to this thesis. 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.

Table 2.6: Comparison of six approaches

CAA approach	CAA behaviour method	Context information impact on CAA	Context source impact on CAA	New UML notations for CAA	CAA structure method
Existing approach	√	×	×	×	√
Masreiter	√	×	×	×	×
Choi	√	×	×	×	×
Almutairi	√	√	×	√	×
CA-UML	√	√	√	√	√
Context-UML	×	√	√	×	√

CHAPTER 2. LITERATURE REVIEW

Researchers have recently used new UML diagrams in their attempts to make CAAs easier within the field of user-environmental interactions. There follow accounts of different modelling forms presented in recent research as extensions of the use case diagram to model CAAs [2, 3, 4, 9, 16, 21, 22, 23, 24].

Table 2.6 presents a comparison of six approaches to the use of an extension of UML to model CAAs: the existing approach, Context-UML, and the approaches proposed by Masreiter, Choi and Almutairi, and *CA-UML*.

Choi and Lee [3] 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 CAAs by UML. They specified context such as temperature ranges to produce actions, although this research is still limited in its ability to express them in the use case diagram or explain how CAAs are affected by contexts and context sources. Additionally, the CAA 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 CAA.

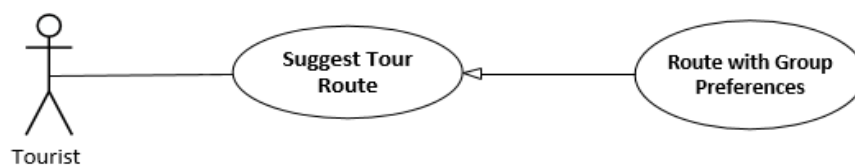


Figure 2.9: An example diagram of the Choi approach

The use case diagram shown in Figure 2.9 does not include real context or illustrate which context source is required to provide a set of context. Furthermore, the research does not mention any context types or how to model atomic context information or composite context informations; these are needed in order to make adaptation actions to allow CAA to be self-adaptive depending on changes to the user's environment.

Almutairi et al. [4] enhanced the use case diagram to model CAA. This paper suggested new shapes used (dotted shapes for actors and use cases) if the use case required other use cases to extend the main use case and to specify context factors such as time and location, although the extended shapes were still limited to

CHAPTER 2. LITERATURE REVIEW

expressing the CAA interactions between context source and context. Also, there is a lack of modelling as a consequence of CAA; the enhancement of the CAA involves complex activities such as research into UML, which needs to be extended by specifying certain CAA behaviours: acquisition sensing and adaptation actions.

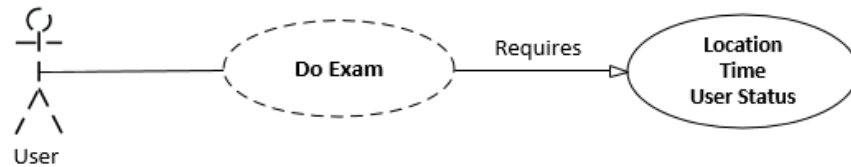


Figure 2.10: An example diagram of the Almutairi approach

The use case diagram represented in Figure 2.10 reveals the limitation of attempts by Almutairi et al. to model the required context behaviour affecting the main use case (Do Exam): this does not investigate how to get context as the main input into CAA, or which context source is needed to provide context. Furthermore, the context cannot be limited to location and time; the use case diagram should consider a wider range of contexts and CSs to demonstrate how they interact with each other.

Masreiter and Metzker [9] tried to depict CAA functionalities and behaviour as well as context situations using existing use case diagram notations; they attempted new processes for context situations, aiming to capture and specify CAA requirements, but they did not identify real situations between user needs and CAA functionalities to express the context information, context source and CAA interactions. Most CAA modelling approaches by UML to model CAA structures are still limited for CAA behaviour; the use case diagram and the activity diagram are the most popular approaches to model any system functionalities or behaviour (use case diagram for system functionalities and activity diagram for system behaviour).

Kang et al. [21, 22] 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 UML 2.0 to design effective models. The authors used

CHAPTER 2. LITERATURE REVIEW

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. In addition, the authors extended the activity diagram to enhance modelling of mobile agents which depict the applications' elements as well as exchanging messages between multi-agents that express the hosts' interactions. Finally, this research has improved this new approach, which is able to use UML elements of the activity diagram to model specific issues of mobile agents.

Alghathbar et al [23] 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.

It is necessary to analyse the design level of security requirements and evaluate the requirements because validating information flow requirements at an early stage will decrease the cost of fixes in the later stages. UML diagrams can show the interaction between diagrams to identify and specify the system behaviour, functionalities and logic, such as use cases' interactions with objects and entities. This paper focused on sequence diagrams and how the interaction will be addressed. There are three parts to the interaction, namely actors, entities and control objects. It will also examine how to establish the security policies for complex messages and exchange information between actors and entities, and the following three instructions to build a complexity message of creation messages (creation objects), iteration messages (send messages and get feedback multiple times) and conditional messages (control messages to be under conditions and rules). Finally, UML analysis will be undertaken, with flow control policies such as MAC (Mandatory Access Control) and DAC (Discretionary Access Control).

Almutairi et al. [4] enhanced the use case diagram to model CAA. The authors created a section for the activity diagram and adjusted the new approach to include new notations for the activity diagram to model CAA, which focused on context categories,

CHAPTER 2. LITERATURE REVIEW

especially when and where to capture context information. The authors suggested three notations for context information Store, Context Links and Functional Requirements.

In the present study, the author investigates an example system using an activity diagram of M-Learning as a CAA and uses a new approach to clarify the activities using both existing and new notations, which require two categories of contexts (time and location). In addition, the main functionality is used to download material, as illustrated by the use case diagram and demonstrated by the activity diagram such as the models of CAA behaviour, especially adaptation activities. Self-adapting processes need to be clarified carefully, but this is not demonstrated in this study. In addition, the CAA output depends on the adaptation processes being a self-adapting system, which is the main objective for any CAA.

Sindre and Opdahl [24] proposed a new concept of misuse cases based on the use case diagram. They focused on three techniques to identify and describe misuse cases, specify the requirements needed by misuse cases at an early stage and develop a test method to verify the security requirements. The purpose of this study is to identify any risk that can arise in the future when the system is at runtime level to protect it against any attacks, because it is cheaper and easier to repair any errors at the analysis level and produce software that is adequate to achieve the objectives and goals of the user and the system. Furthermore, the misuse cases method describes the system functionalities to provide high level security for the system.

Almutairi et al. [16] published a paper on Specifying Security Requirements for Context Aware Systems using UML. They adjusted their new approach using a use case diagram to model the functionalities of CAA.

In this paper, the authors specified security requirements for CAA. Any system has functional requirements and non-functional requirements, which differ in that the system needs functional requirements to be run and implemented which are not required for security, reliability and so on, but the system also needs non-functional requirements for quality and privacy.

CHAPTER 2. LITERATURE REVIEW

Many researchers disagree about security requirements and most of them consider that security is a non-functional requirement. Some consider that security should be a functional requirement before system implementation and configuration.

Sheng and Benatallah [2] used the Context-UML approach of class diagram to describe the development of the initial stage of a complete model-driven approach for context awareness services: they modelled CAAs using Context-UML and their implementations were automatically generated and then deployed. Furthermore, the Context-UML approach investigated users requiring the provision of the web or more intelligent services to understand their current personal situation. CAA developers also implement anything relating to context management, such as the collection, dissemination (spreading) and usage of context information. The transferability of service design has been developed as a result of the technical independency of service models. The service models can be integrated into new technologies such as new web services by basically developing new transformation rules for an active model.

The Context-UML Approach has tried to resolve the lack of formalizing of the development of CAAs. As a result of this, the development of CAAs is difficult and time consuming, especially in complex cases. The Context-UML Approach uses stereotypes to simplify and represent system requirements. But the authors do not model acquisition or context information and the context services that are abstracted from Context-UML encompass sensor details, which in turn provide information through interpreting and developing the sensed information (such as raw context information). The authors do not investigate context source and disguise the complexity of context acquisition from CAAs' designers, focusing on the functionalities of CAAs, rather than on context sensing. In Context-UML, it is not necessary for CAAs' designers to specify which context services are needed for context information retrieval at the design stage; such a decision regarding which context service is selected for the provision of a context is postponed until the invocation of CAAs.

In other words, the stereotypes approach is a special mechanism to classify CAA structural elements and divide CAA modelling into context modelling and context

CHAPTER 2. LITERATURE REVIEW

awareness modelling. The context modelling classification outlines the input parameters of contexts and their providers to be carried to CAA, and also specifies the relationships between them. Context awareness modelling outlines special functions to implement CAA requirements and their constraints to make adaptations of results, as well as to exchange CAAs' entities and their values.

2.7 Summary

This chapter has provided a general overview of RE, UML and CAA and their important aspects. Requirements engineering of CAA is an interesting field for researchers using different approaches. This chapter has described the differences between important aspects of requirements engineering such as requirements engineering processes, categories, types and approaches, and then outlined the modelling figures and why CAA needs simulation approaches, as well as identifying the static and dynamic aspects of UML and their limitations.

This chapter has also investigated the importance of existing UML diagrams and the practical usage of their notations to depict CAAs' functionalities, behaviour and structure using UML diagrams and investigated the limitations and weaknesses of these notations. This indicates why contexts and CSs are needed for CAAs and why their changes always affect CAAs based on environmental changes around the user and his/her preferences. Moreover, this chapter has defined the main CAA components and identified context source and context information, which are the most important aspects for intelligent systems such as CAAs.

This chapter has also reviewed the existing approaches using UML diagrams to model CAAs' functional or non-functional requirements and outlined their research limitations for depicting the behaviour and structure of CAA requirements.

The aim of this thesis is to extend UML diagrams to resolve the limitations of existing UML approaches to enable them to fulfil the behaviour and structure of CAAs'

CHAPTER 2. LITERATURE REVIEW

requirements. The approach of CA-UML diagrams will also be compared with other existing approaches to find the best solutions for CAA as a complex system and understand its needs through simulations and transforming everything into available information that can be used to provide real actions and services.

In the following chapter, this thesis suggests a new notion – the context aware use case diagram – to model the functionalities of context aware applications.

3. An Extension of the Use Case Diagram to Model Context Aware Applications

Objectives:

- Extend the UML use case diagram to depict the functionalities of CAAs;
- Adjust the existing notations of the use case diagram to model CAAs' functional requirements;
- Propose a new concept of the use context diagram to model CAAs' context-awareness requirements;
- Suggest new notations to model use context, context source and their relationships;
- Design a standard modelling for CAAs' functionalities;
- Specify the functional requirements and context-awareness requirements for CAAs using a new notion known as the context aware use-case diagram;
- Extend the UML usage scenario template to describe the functionalities of CAAs and suggest a new concept of the context aware usage scenario template for the context-aware use case diagram notion.

3.1 Introduction

CAA has become the most interesting technology using smart software and hardware in real life, and UML requires further enhancement so that it is suitable for CAA requirements. Currently there is insufficient modelling of UML use case diagrams for CAAs used in describing the dynamic of CAAs' functionalities, CI and CS through use case diagram notations. Use case diagrams are still limited to modelling the behaviour of CAAs, identifying which CI is required for CAAs and expressing the interactions of contexts with CSs [9, 27, 28, 120]. More importantly, this chapter applies the requirements engineering aspects to specify the functional requirements and the context-awareness requirements which provide the developers with a clear understanding of the end user's requirements and what contexts the system must be aware of.

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

The main objective of this chapter is to propose an extension notion of a context aware use case diagram to clarify the requirements of CAA and reduce the complexity in the visualization of CAAs to support the specific user needs and the understanding of users' intentions, depending on changes in the user environment [1]. For example, if a driver uses a mobile device while driving, this results in many instances of CI, such as location, speed, weather conditions and other contextual factors.

There is a new class of applications that are aware of the CI environment. Such CAAs adapt according to CI values as well as context changes over a period of time [14]. In addition, CAAs are also able to check the computing environment to react to environmental changes [8, 18].

The context-aware use case diagram notion is used to model CAAs by means of a use context diagram to illustrate their context-awareness requirements and their behaviour. This chapter will describe how CAAs may be presented, and also specify the functional requirements of CAAs through the basic notations of the use case diagram. This notion merges practical examples of existing notations with new notations for the use context diagram to create clear models for CAAs. Context-awareness has three requirements of context information as follows:

- Sensing the object's properties and extracting information from its environment;
- Monitoring changes to context information and their reasons;
- Responding to the context information changes and adapting in time.

Moreover, this chapter investigates the requirements that have continuously changing results based on continuously changing inputs to represent those changes in use contexts using rich examples to represent the practical usage of new notations. It follows from these context requirements that the user's location, the user's activity, who the user is with, the state of the physical environment (such as time, temperature, weather, and noise level), and nearby resources are important aspects of context [80, 89].

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

In addition, this chapter provides a high level modelling of CAAs, which abstracts the main functional requirements and context awareness requirements and outlines simplified examples that express various models using standard modelling to illustrate the new notations and their goals. Furthermore, this chapter investigates use context diagram results in a rich and expressive language of modelling and analysis for CAAs [58].

3.2 Traditional Use Case diagram

UML is a diagram language which enables designers of information systems to outline the system behaviour and structure. However, UML diagrams possess static modelling and dynamic modelling; such diagrams can also provide an abstract of system modelling to illustrate high level system functionalities, such as the use case diagram, or provide details of system modelling to demonstrate low level system activities, such as the activity diagram [6, 10].

There are thirteen different diagrams in the UML for modelling the static and dynamic aspects of systems [17]; the use case diagram is one of them, and is used for dynamic modelling. They are central to modelling the behaviour of an application and represent an efficient means of visualizing, specifying and documenting the intended behaviour of an application during the gathering of requirements and analysis.

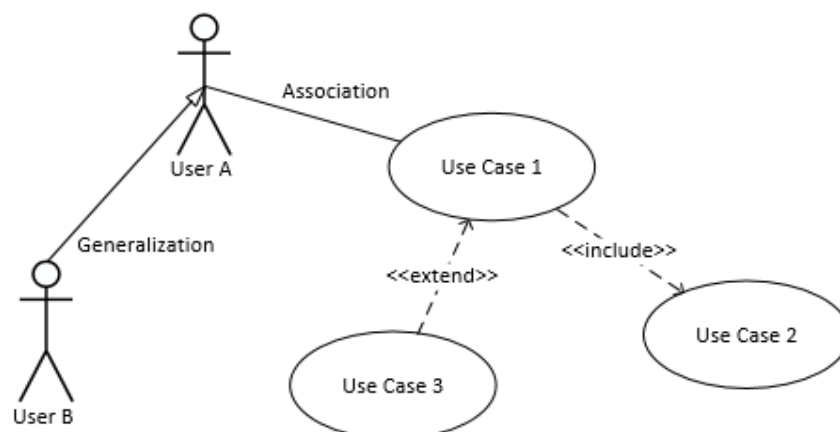


Figure 3.1: Traditional use case diagram notations and their relationships

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

The diagram shown in Figure 3.1 describes the basic notations used for use case diagrams; following this, the most important notations are the actor and the use case. We can observe that the use case shows that one task can be represented for a normal system, but it does not have enough notations for CAA functionalities or behaviour. The use case diagram is built from three basic notations: the use case, the actor and the relationships between them, as shown in Figure 3.1.

Use case notations may be divided into three types as follows:

- Use case notation

Use cases apply the system functions used in UML modelling to capture the system's functional requirements. A use case notation describes the system's behaviour in general or partial terms as well as presenting the system functionalities (such as what the system or part of the system can do), without describing system behaviour in the form of use cases and how that behaviour is to be implemented. In addition, use cases provide an effective way for developers to communicate with the application's end user and domain experts during the requirements gathering and analysis phases of the software development lifecycle [9, 27, 28].

Furthermore, use cases are used for validation and testing during application development. A use case has a name and is graphically rendered as an ellipse, as depicted in Figure 3.1.

- Actor notation

Generally, the use case diagram defines the actor and his actions with system functionalities. An actor notation represents a coherent set of roles that users of use cases play when interacting with these use cases [17]. Actors can be human, external hardware or 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, as in Figure 3.1.

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

- Relationships

Typically, use case diagrams gain a set of relationships connecting the actors and use cases with each other and controlling the behaviour of system functions. There are three kinds of existing relationships between the notations of the use case diagram as follows:

- Generalization Relationship

A generalization relationship is a relationship between two or more actors (or between two or more use cases) which means that the child actor can inherit the behaviour and the meaning of the parent actor; the child may add to or override the behaviour of its parent; and the child may be substituted at any point where the parent occurs [5]. The generalization relationship is represented graphically as a solid directed line with a large open arrowhead. For example, in Figure 3.1, “actor B” is a generalization of “actor A”.

- Association Relationship

An association relationship is represented graphically as a solid directed line without an arrow, connecting the actor with the use case. Usually, an association relationship specifies the actor requirements (such as the user or external system); this relationship also indicates the functional requirements needed for the actor. For example, in Figure 3.1, "actor A" interacts with a system function, "use case 1".

- Dependency Relationship

A dependency relationship has two types of use cases relationship: <<include>> for mandatory functional requirements and <<extend>> for optional functional requirements. An <<include>> relationship between use cases means that the base use case explicitly incorporates the behaviour of another use case, while an <<extend>> relationship between use cases means that the base use case implicitly incorporates the behaviour of another use case. Graphically, both relationships are rendered as dependencies, stereotyped as <<include>> and <<extend>> respectively. For example, in Figure 3.1, the base use case “use case 1” extends “use case 3”, while the base use case “use case 1” includes “use case 2”.

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

3.3 CAAs Functionalities Modelling

Modelling the functionalities of CAA using existing notations of the use case diagram and suggesting new notations of the use context diagram can create different modelling scenarios which support CAA designers and are aware of all environment factors that might affect the user and specify the functions of CAA. This notion also makes it possible to outline which CS are needed and their availability. The difference between the traditional system and CAA is that the traditional system will not perform any action until a request is made by the user; also, the input parameters may be inserted by the user or stored in the system database, whereas in CAA, the user can only specify the personal preferences that can control system services; then the system will capture the parameters needed to be a self-adaptive system, depending on changes in the surrounding user environment.

Table 3.1: Comparison between traditional use case diagram, adjustable use case diagram and use context diagram

Concept Name	System functional requirements	CAAs functional requirements	Context-awareness requirements
Traditional Use Case diagram	√	×	×
Adjustable Use Case diagram	√	√	×
Use Context diagram	√	√	√

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

The following sections outline two concepts to be merged for the proposed extension of the context aware use case diagram as follows:

- Adjusting the notations of the Use Case diagram for CAAs modelling;
- Creating the notations of the Use Context diagram for CAAs modelling.

3.3.1 Adjusting the notations of the Use Case diagram for CAAs modelling

The basic notations can be adjusted for CAAs' functional modelling, which can represent the functional requirements of CAAs. However, Use Case diagram notations are still limited to depicting the context awareness requirements, and extension is needed to include more notations to enable the functionalities modelling of CAAs. The existing notations of the use case diagram are still able to express the behaviour of CAAs' functional requirements, as modelled in Figure 3.2.

In fact, we can use the same notations that were previously used in the traditional use case diagram to describe CAAs' functional requirements; such CAA functions need only a use context representing Atomic CI (Atomic CI is a single CI represented by a use context notation for CAA); or the CAA function needs a set of Atomic CIs (many Atomic CIs are mandatory to make Composite CI which are represented by the use case notation for CAA). Furthermore, CAAs' functional requirements change depending on user requirements or CAA needs, while CAA context awareness requirements change in relation to the status of the objects sensed (Contexts), which affect the CAA's behaviour depending on changes in the user's environment.

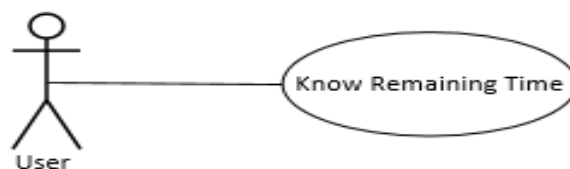


Figure 3.2: An example of adjustable use case diagram notations

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

To support this, the example shown in Figure 3.2 depicts a functional requirement of a navigation application using the existing notations of the use case diagram but adjusting their usage to represent Composite CI (know remaining time), which can show the functional requirements of CAAs. However, it is still unable to represent context awareness requirements; in addition, use case diagram notations represent normal system functional requirements or CAAs' functional requirements, but existing notations need new notations to model the context awareness requirements of CAAs.

The following diagrams will express the importance of adjusting the existing notations of the use case diagram as well as creating new notations for the use context diagram and relationships which are specified for the modelling of CAAs' functionalities. The use case diagram represented in Figure 3.2 shows an example of Composite CI expressing the most important functional requirement of the navigation application. However, the basic use case notations for normal system functionality are suitable to be the Composite CI notation in the concept of the adjustable use case diagram.




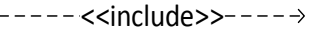

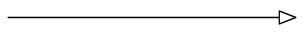
In other words, this means that CAAs do not require additional notation for Composite CI and can use the notations of the use case diagram to specify the CAA functional requirements and link them with new notations provided by the use context diagram to achieve comprehensive modelling for CAA. Composite CI is a high-level CI that needs to interact with at least two instances of Atomic CI by special calculation to understand the required CI and may not interact directly with CSs. Atomic CI is a low-level CI that does not interact with another Atomic CI and can interact with CSs.

Furthermore, Figure 3.2 shows an example function of <Know Remaining Time>; this is not a traditional system function and requires other notation activities to visualize CAA functionalities clearly and accurately.

It also depicts the most important function for the Navigation application (the <know remaining time> function requires two Atomic CIs of speed and distance). Table 3.2 shows the adjustable usage of existing notations of the Use Case diagram.

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

Table 3.2: Adjustable notations of use case diagram for CAAs

Notation	Traditional Usage	Adjustable Usage
 Actor	Visualize the person or the external system who interacts with the system	Visualize the person who needs to be aware of contexts by interacting with CAAs
 Use case	Depict the functional requirement provided by the system	Depict the functional requirement provided by the CAAs
 Association	Actor and Use Case relationship for system functions modelling	Actor and Use Case relationship for CAAs functions modelling
 Include relationship	Use Cases relationship for mandatory system functions	Use Cases relationship for mandatory CAAs' functions
 Extend relationship	Use Cases relationship for optional system functions	Use Cases relationship for optional CAAs functions
 Generalization relationship	Actors' relationship for system entities	Actors' relationship for CAAs entities





The following section investigates the importance of the new concept of the use context diagram, including more notations and relationships specified in the context awareness requirements for CAA modelling.

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

3.3.2 Creating the notations for the Use Context diagram for CAAs modelling

New notations are necessary to represent CAA behaviour and CI is the main parameter affecting the behaviour of CAA. By analogy to the use case diagram, this chapter introduces the concept of the use context diagram to model the acquisition, the aggregation and the inference of CIs relevant to CAAs. 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 CI that affects the behaviour of the application. Furthermore, different notations are provided to control CAA behaviour by tracing the availability of CIs and tracing their available providers.

Table 3.3: New notations of the Use Context diagram for CAAs

Notation	CAAs Usage
 Context source	Visualize the sensor, context service or web service that creates context information and interacts with CAAs
 Use context	Depicts context information which is needed for CAAs to represent the context-awareness requirements
 Context association	Relationship between context source and use context
 Include relationship	Use contexts relationship for mandatory context-awareness requirements of CAAs

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

-----<<extend>>-----> Extend relationship	Use contexts relationship for optional context-awareness requirements of CAAs
--	---

A use context diagram is composed of a set of use contexts and CSs and the relationships between them, as depicted in Figure 3.3.A.

The above Table (Table 3.3) of new notations for the Use Context diagram is divided into three types depicting the context-awareness requirements as follows:

- Use Context notation

Depicting use contexts to specify which CI are needed for CAA, which expresses the objects' situations. This also helps to identify the reasons affecting CIs' status depending on the user environment, which changes continuously, especially through the run-time of CAA. In addition, the use context specifies the CIs that affect the behaviour of an application or part of an application. It is a description of a set or 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 behaviour of the application under development, without having to specify how the measurement of those CIs is actually implemented. They also provide the developers with 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, they are excellent sources of integration and system

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

tests. A use context has a name and is graphically rendered as a dotted ellipse, as in Figure 3.3.A.

- Context Source notation

CS notation represents the provider of CI which senses the object (transforming the sensed information) and specifies the acquisition behaviour, which enables CI to be captured from the user environment and carried to such applications and to adapt their changes; CS can be internal systems within the scope of a CAA, such as embedded systems [2], or external CS, such as an external sensor or web server, which would normally be outside the scope of the CAA [13].

In the context-aware use case diagram notion, CS infrastructure is not required in more detail and CAA designers need to know which CS is providing CI; this notion should reduce the CS's complexity of CI sensing for system developers, so that they can easily specify which functionalities of CAA are required [2, 5, 8].

CSs are to use contexts what actors are to use cases. Use contexts communicate with CSs to gather raw context data from which CIs are calculated.

Typically, CSs 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). Graphically, they are rendered as shown in Figure 3.3.A.

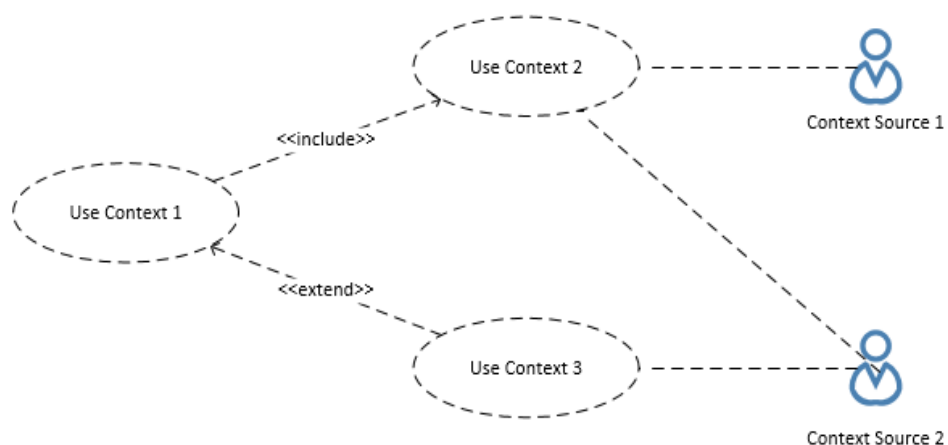


Figure 3.3.A: An illustration of the notations for use context diagrams

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

- Relationships

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. In addition, Use Context diagrams connect use cases, use contexts and CSs by a set of relationships as follows:

- Context Association Relationship

CSs 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 CAAs, there will be use contexts that are specialized versions 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. An example of the context association relationship between the CS and the use context is given in Figure 3.3.B.

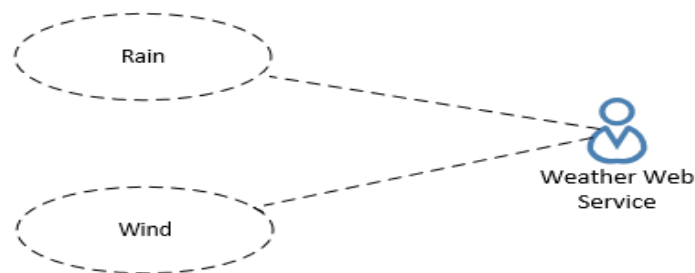


Figure 3.3.B: An example of a context association relationship

- “Include” Relationship

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 example of the “include” relationship between the use context “weather condition” and the use contexts “location”, “temperature” and “rain” is given in Figure 3.3.C. This means that the location, the temperature value and the rain status are mandatory CIs to be included in the weather reports.

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

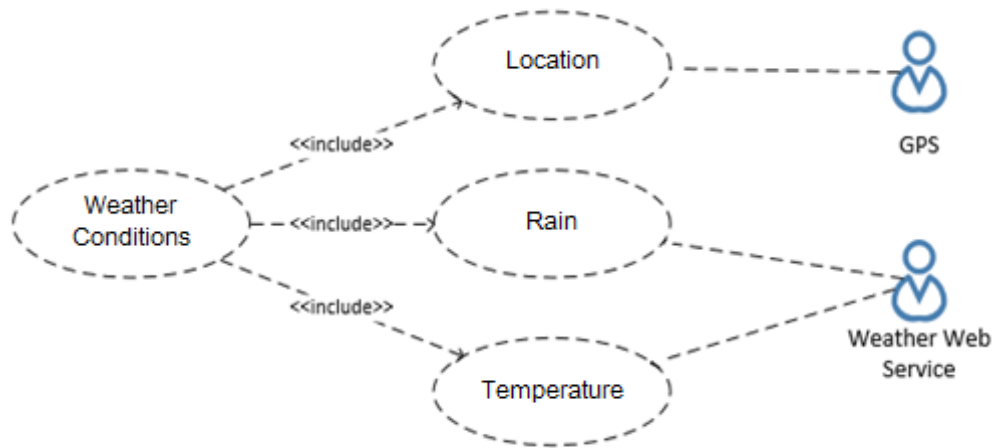


Figure 3.3.C: An example of an “include” relationship

- “Extend” Relationship

An “extend” relationship is used to model the part of a use context that the user may see as optional CI. In this way, optional CIs are separated from mandatory ones. Figure 3.3.D shows an example of a 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.

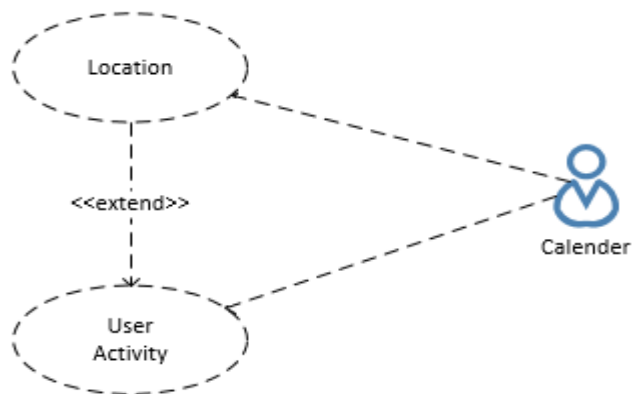


Figure 3.3.D: An example of an “extend” relationship

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

3.3.3 Context Aware Usage Scenario Template

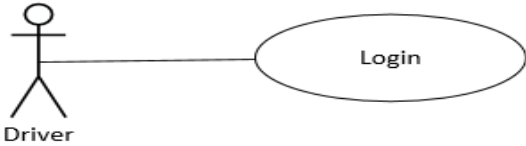
Usage scenario templates are able to describe the functional requirements for all kinds of software applications and systems as well as the functional requirements of CAAs to provide sufficient details for each use case to specify the information needed in the architecture stage. Typically, usage scenario templates explain the interactions between users and systems in sequence sentences as stories to outline the users' selections and system actions. However, usage scenario templates need to be extended for CAAs to include a third party for interactions to be described by users, CAAs and CSs to specify the exact actions that should be made by CAAs depending on user preferences and the CIs captured from CSs. The CA-UML approach suggests an extension of the usage scenario template called the *Context Aware Usage Scenario Template* to include use contexts which express different scenarios of CAAs. In other words, the context-aware usage scenario template merges existing use case scenarios and use context scenarios to describe the CAAs' stories in clear steps using English sentences. The developers of CAAs need to use context-aware usage scenario templates to understand the specific CAAs requirements of functional, non-functional and context awareness.

Misunderstanding of CAAs' functionalities will cause real problems in later stages of CAAs' life-cycle development and the quality services of CAAs depend on this level to be successful without a loss of time and budget. Context-aware usage scenario templates will create clear documentation and project management for CAAs. Context-aware usage scenario templates aim to define many scenarios for CAAs' services by writing a story of CAAs' scope and controlling the integration between users, CAA and CS as work-flows of use cases and use contexts. In addition, the context-aware usage scenario template helps to move from the context-aware use case diagram notion to the following notion of the context-aware activity diagram. In other words, the context-aware use case diagram notion models the functionalities of CAAs in diagrams and the context-aware usage scenario template concept describes the context-aware use case diagram in clear words as stories. Furthermore, the context-aware usage scenario template concept clearly documents CAAs' functionalities and extends the elements of the basic usage scenario template.

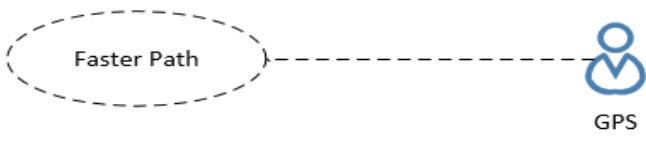
CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

An example of a navigation system using the context-aware usage scenario template concept for the use case of <login> and the use context of <know faster path> is shown in Table 3.4. It involves the same aspects of the use case as a scenario and new aspects created for the use context scenario, which are merged together to describe all steps, use case conditions, use context constraints and other details for the navigation system.

Table 3.4: An example of a context aware usage scenario template

Elements	Example
Use Case Name	Login
Use Context Name	Faster Path
Use Case ID	Give a unique number for each use case for functional requirements and give a unique letter for each use case for non-functional requirements
Use Context ID	Give a unique capital letter and number for each use context for context awareness requirements
Use Case Diagram	

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

<p>Use Context Diagram</p>	 <p>The diagram shows a use case labeled 'Faster Path' enclosed in a dashed oval. A dashed line connects this use case to an actor icon labeled 'GPS'.</p>
<p>Use Case Priority</p>	<p>High</p>
<p>Use Context Priority</p>	<p>Medium</p>
<p>Use Case Scenario narrative</p>	<ul style="list-style-type: none"> - The driver uses the navigation system interface to log in - The driver inserts a valid user-name and password in login - Navigation system should verify the driver's login information - If the driver inserts correct login information - Navigation system allows the authorized driver to access - Navigation system will display the main screen - The driver can select an available service - Else the system exits the interface screen
<p>Use Context Scenario narrative</p>	<ul style="list-style-type: none"> - The driver inserts the user preferences such as postcode - Navigation system retrieves the maps for the driver - Navigation system shows options of paths for the driver - Navigation suggests the faster path - The driver selects the faster path

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

Actor	Authorized driver
Use Case conditions	Stakeholder must be an authorized user and have a valid account, and must update the account every month
Use Context Constraints	Cancel the journey direction of faster path if a road is closed or there is a traffic jam and notify the driver of any accidents on the road
Frequency	Once per visit
Awareness Requirements	Update driver of future events during journey time, such as road closures, accidents and heavy traffic
Context Source Name	GPS
Alternative CS Name	None

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.

3.4 A Context Aware Use-case Diagram

This section outlines an extension notion called the context-aware use case diagram, which merges the existing notations of the Use Case diagram with the new notations of

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

the Use Context diagram to address CAAs' modelling. In addition, this notion is more expressive to ensure clear modelling of the functionalities of CAAs: this enables a clear separation of concerns between context-awareness requirements and functional requirements. Modelling CI, which refers to context as a dotted notation in the CA-UML approach, serves to support CAAs' modelling, simulation and documentation. Depicting CAAs' requirements for CI and relevant CS properties enhances the CAA architecture when using special notations to express CAA behaviour and structure; CI notation in CA-UML also means that CI is sensed by CS.

CI can be categorized into two types as follows:

- Atomic Context Information
- Composite Context Information

A more important issue for the context-aware use case diagram notion is a new relationship between use cases and use contexts for retrieving CI values.

- Utilize Relationship







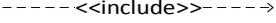

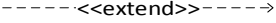




A context-aware use case diagram is built from a set of use case diagrams and use context diagrams by linking use cases to use contexts using *utilize* relationships. A *utilize* relationship between a use case and use context means that the behaviours specified by the use case depend upon the CIs described by the use context. A *utilize* relationship is graphically rendered as a dependency, stereotyped as <<utilize>>, as in Figure 3.4. 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 3.4.

Examples of context-aware use case diagrams are detailed in the following sections.

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

Table 3.5: 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	

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

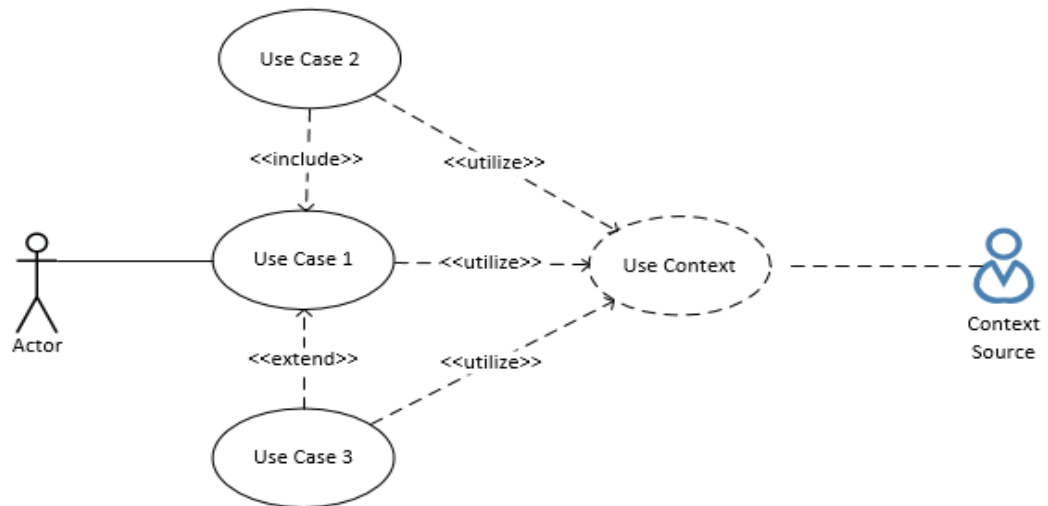


Figure 3.4: An illustration of the notations for a context-aware use case diagram

3.4.1 Applicable capturing methods by Context Source

CS has the ability to sense an object and carry it to an application, and might also capture CI from sharing web services. There are three applicable capturing methods by which CS can interact with environmental objects to read CIs and their parameters as follows:

- One-to-one capturing: a single CS provides a single CI for context-awareness requirements.

A concept of the Use Context diagram representing new notations to specify the real behaviour for CAA which allows expression of the capturing method through notations and specifies CI for each use context. To support this capturing method for one-to-one capturing by CS (CS-to-CI), the example shown in Figure 3.5.A specifies the practical usage of proposed notations which describe CS as a notation and their services. Atomic CI is the most important service produced by CS and is required for CAA.

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

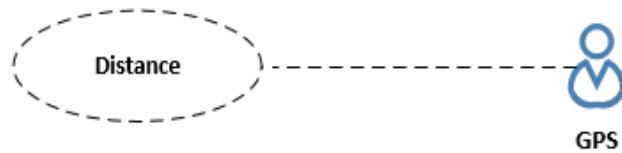


Figure 3.5.A: An example of one-to-one capturing method

This capturing method also examines the context-awareness requirement in Figure 3.5.A. a single CI provided by CS means that the CI type is Atomic CI.

The following diagrams and examples will describe the usage of new notations in more detail.

- One-to-many capturing: a single CS provides multi-CIs for context-awareness requirements.

This method depicts the ability of CS to create a set of CIs needed for a single or multi context-awareness requirement.

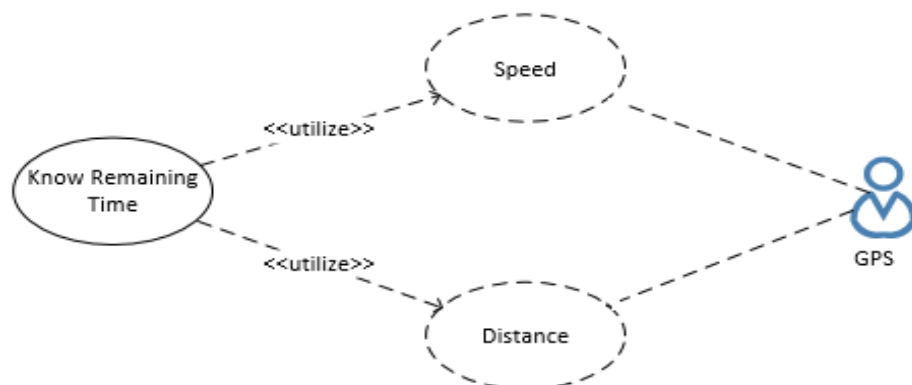


Figure 3.5.B: An example of one-to-many capturing method

Figure 3.5.B shows an example of this method, depicting the two use contexts required for a functional requirement. Remaining time is a functional requirement of the navigation application (an example of CAA) and is visualized by the Adjustable Use Case diagram expressing Composite CI, which needs two Atomic CIs of speed and distance.

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

In addition, this method shows how the response to the Composite CI needs to interact with different Atomic CIs to calculate the required value of Composite CI to deal with user functional requirements and context-awareness requirements for CAA.

Moreover, it shows that the Composite CI indicates the remaining time until the final destination point has been reached; also, remaining time is calculated using two Atomic CIs of speed and distance.

- Many-to-one capturing: multi-CSs provide a single CI for context-awareness requirements.

For different reasons, such as CS availability or other issues, this method is important to keep retrieving CI and their changes via a set of CSs. However, this method shows how one or more CSs can provide a single Atomic CI; therefore, it is important that one Atomic CI is captured by different CSs, in order to connect with the fastest CS. This does not mean that the CI type is Composite CI; it is still an Atomic CI that can be produced by CS or interact directly with CS web services.

To support this method, in Figure 3.5.C, an example using Use Context diagram notations shows how such a location can be sensed by GPS or a web service such as the *WhereAmI* service. This flexibility of connection with many CSs will increase availability to any time or place.

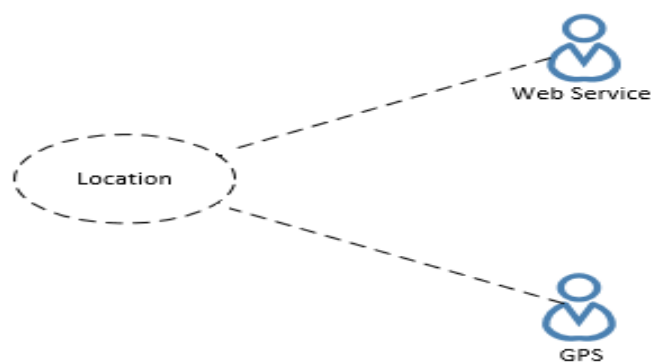


Figure 3.5.C: An example of many-to-one capturing method

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

3.4.2 A standard Context Aware Use-case diagram

The context-aware use case diagram notion provides a standard modelling for CAAs modelling which merges two concepts' notations and their relationships to depict the functional requirements and context-awareness requirements.

CAAs' functionalities modelling needs to determine the acquisition of CS and details of the CI by interpreting the sensed information; CAA designers can specify context-awareness requirements and which CSs are needed for CIs retrieval at different stages of the CAA development life cycle. CAA analysts have different models that show the most accurate functionalities of CAAs using different approaches; in fact, CAAs are complex and have different boundaries that interact with each other, and for this, a standard modelling is necessary which enhances the functionalities modelling stage of CAAs [2, 6]. Furthermore, the main issue in a context-aware use case diagram notion is representing a set of notations to establish whether the CI is of the Atomic CI or the Composite CI type.

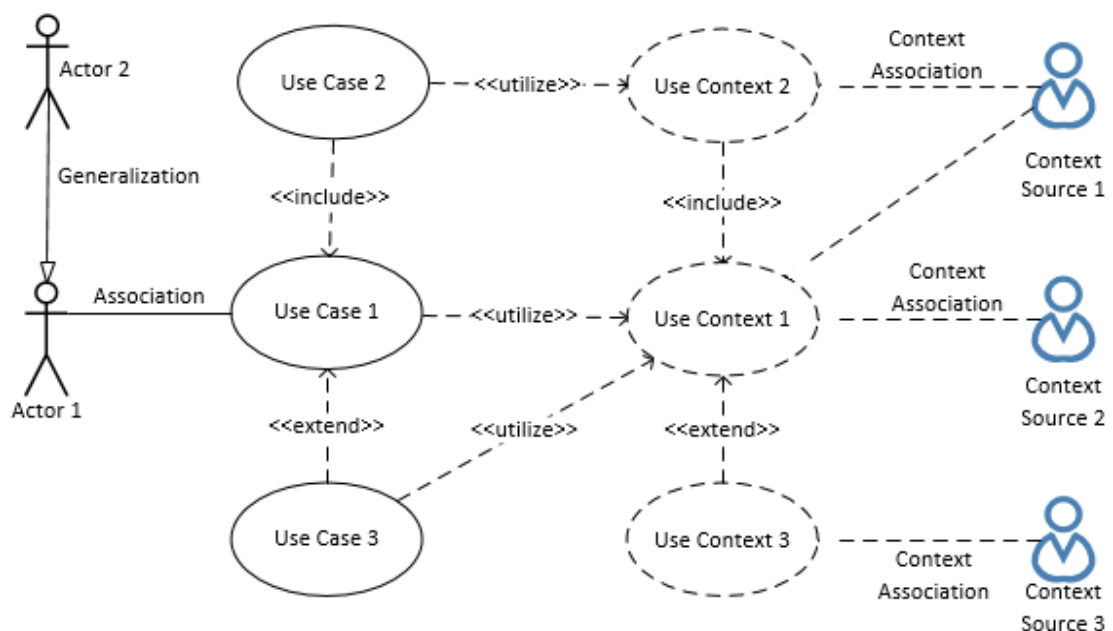


Figure 3.6: A standard modelling of a context-aware use case diagram for CAAs

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

The standard modelling is depicted using the adjustable notations of the Use Case diagram merged with the new extension concept of the Use Context diagram. The two concepts are connected by the new *Utilize* relationship to provide the new notion of a context-aware use case diagram for CAA functionalities modelling.

In Figure 3.6, we can observe that the CSs are outside the scope of CAA functionalities. However, users cannot specify which CSs are needed for CI retrieval to fulfil the context-awareness requirements; interaction between systems is only possible by calling each other, such as XML messaging with web services or external sensors (sharing services can be connected with systems or sensors) [1, 2].

In addition, the context-aware use case diagram modelling begins the modelling stage using a type concept of Use Case diagram notations which presents the functional requirements of CAAs, is easy to understand and makes it clear which software functionalities and plans are appropriate; it is also necessary to know how the applications interact with each other [9].

For a clearer understanding of the context-aware use case diagram notion for CAAs, this thesis considers real-life case studies of CAAs in Chapter 6 and 7.

3.4.3 Specifying the functional requirements for CAAs using a Use Case diagram

When representing models using the basic notations of the Use Case diagram, the difference between system modelling of functional requirements and CAA modelling of functional requirements is that the system functions are completed and ready for the design stage but the CAA functions still need more notations to represent the context-awareness requirements, which affect the functional requirements' behaviour.

However, using existing notations of the Use Case Diagram, it is possible to model the functional requirements of CAA, but each notation is adjusted for other behaviour activities, such as actors representing the user or external system, which applies with

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

CAA functions and the use case representing a single function which needs a single Atomic CI or may represent a composite function. This requires a special method to calculate the values of Composite CI, which needs a set of Atomic CIs to fulfil CAAs' requirements.

3.4.4 Specifying the context-awareness requirements for CAAs using a Use Context diagram

Today's mobile devices use the internet to connect everyone with everything everywhere; if CAAs are to be applicable to meet Ubicomp requirements, they need to have full Cloud computing environmental services to cover all context-awareness requirements using a set of context services to capture CIs and carry them to applications.

In other words, context-awareness requirements are completing the functional requirement of CAA. In addition, functional requirements may provide a single function which needs a single Atomic CI (Atomic CI fulfils the context-awareness requirement); or alternatively a composite function (as fulfilled by Composite CI); this means that in the context-aware use case diagram notion, a composite CI is a high level CI which is not a direct counterpart of the CS and needs multiple Atomic CIs, or possibly other Composite CIs. Furthermore, in the context-aware use case diagram notion, information about objects sensed from the user's environment is CI, which is categorized into Atomic or Composite information. Multiple Atomic CIs create Composite CI and both fulfil the CAA functional requirements and context-awareness requirements.

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

3.4.5 A practical example of a driving control system using a context-aware use case diagram

Driving Control Systems have become important and interesting in many research institutes to resolve a range of real-life challenges, and their main benefit is in saving people's lives. Thousands of people are killed and millions are injured in road accidents and vehicle crashes every year around the world. The main reason for these accidents is drivers' errors and mistakes. This section presents a practical example of a driving control system. Traditional use case diagrams are still facing challenging issues in terms of modelling smart systems such as driving control systems. Although existing modelling approaches and tools do not meet all context awareness requirements, CAA developers should use a set of concepts and tools to help to execute related elements of CAAs and establish a comprehensive approach for the development life-cycle of CAAs, which is still poor.

Mapping and visualization for the context-awareness requirements should be carefully specified to save people's lives and to guarantee access to systems that are relevant to their needs. A driving control system is a critical system and the error percentage must be 0%. It needs a set of internal and external CSs (internal CS inside the vehicle and external CS outside the vehicle). In addition, this research illustrates how the proposed approach of CA-UML and their diagrams can be used in practice. Sufficient models of context-aware use case diagrams describe the functionalities of CAAs as well as handling their needs in terms of CIs and CSs.

This example investigates driving behaviour and models the main functionalities and their behaviours using the suggested notions of the context aware use case diagram, as shown in Figure 3.7. This chapter considers an example involving the services of a driving control system expressing the context-aware use case diagram notion of adjusted and extended notations to demonstrate their effectiveness. To support the proposed approach of CA-UML, a driving control system is shown in the following diagram to depict the functional requirements and context awareness requirements and merge them to show driving control system functionalities which represent the

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

functional requirements in use case notations and context-awareness requirements in use context notations. In addition, this modelling of a driving control system is an example of how a CAA may be run by specific devices in and around the vehicle, such as roads and other vehicles.

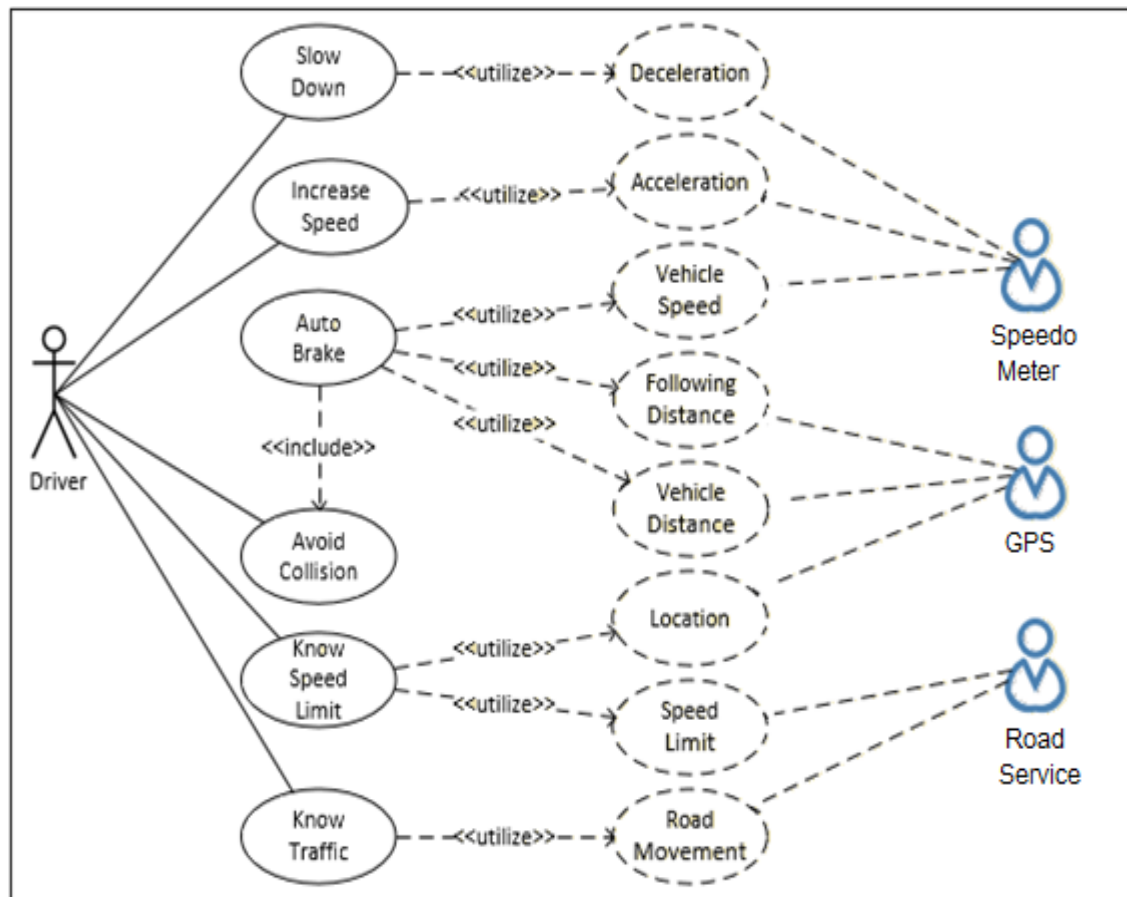


Figure 3.7: Driving control system functionalities

3.5 Summary

This chapter has presented a new extension of the Use Case diagram of UML to fulfil the requirements of CAAs' modelling to be able to depict both functional requirements and context-awareness requirements. This proposed extension is called a context-aware use case diagram. More importantly, this chapter outlines the practical

CHAPTER 3. AN EXTENSION OF THE USE CASE DIAGRAM TO MODEL CONTEXT AWARE APPLICATIONS

usage of a new concept, namely the Use Context diagram, which proposes to represent a set of new notations as well as depicting the providers of CIs (context sources), use contexts and their relationships. In addition, this chapter has investigated how the existing notations of the use case diagram (which describes the application's functional requirements) can be merged with new notations of the Use Context diagram (which specifies the CAAs' behaviour and their context-awareness requirements) via practical usages and examples to model the functional requirements and context-awareness requirements of CAAs.

The concept of use context was introduced to specify the context-awareness requirements and their CIs. Then the concept of the use context diagram was 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. Furthermore, this chapter has filled the gap in CAAs modelling using existing Use Case notations and proposed new notations which provide a powerful visualization to enhance the CAAs dynamic modelling to be able to respond to context-awareness requirements.

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 behaviours specified by the use case depend upon the CIs described by the use context. Furthermore, this chapter has investigated an example of a driving control system to support the merging of the two concepts for the context-aware use case diagram notion. The pragmatics and flexibility of the proposed approach are illustrated using examples of CAAs. In the following chapter, this thesis suggests a new notion, namely the context aware activity diagram, to model the behaviour of context aware applications.

4. An Extension of the Activity Diagram to Model the Behaviour of Context Aware Applications

Objectives:

- Extend the UML Activity diagram to depict the dynamic behaviour of CAAs;
- Adjust the existing notations of the Activity diagram to model the activities of CAAs and context acquisition;
- Propose a new concept of the Context Activity diagram to model the adaptation activities of CAAs;
- Suggest new notations to model CAAs' adaptation activities and constraints;
- Suggest a new swimlane for CS activity flows and meta-swimlanes to express the interaction activities within each swimlane;
- Design a standard modelling for CAAs' behaviour, which illustrates both the high-level and low-level behaviour of CAAs;
- Specify a new notion known as the Context Aware Activity Diagram, merging the existing notations for activities' flows of CAAs and context acquisition with new notations for activities' flows of adaptation activities and their constraints;
- Use the context-aware activity diagram to demonstrate each use case scenario and their functionalities, which are illustrated by the context-aware use case diagram in the previous chapter.

4.1 Introduction

In this decade, many researchers are investigating the needs of self-adapting systems and their life-cycle development, which has faced many challenges.

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

Assistive technologies provide important services such as monitoring and tracking using sets of sensors. Monitoring activities of smart applications control CAAs' behaviour to manage the final adaptation actions for users [81, 86].

Understanding system behaviour is a common challenge for many specialists in system modelling and simulation, especially with regard to complex systems such as CAAs [39]. Simulating dynamic CAAs using new modelling approaches enhances CAA designers' awareness of the behaviour of CAAs and their integration with all software and hardware [21, 23]. UML activity diagrams are still limited to depicting the real behaviour of CAAs and their components, processes and actions, such as the adaptation of output and context acquisition [22]. The main problem for the behaviour of CAAs' modelling is the changes in CI, connection attributes, CS sensing and the user environment, which are in constant flux [21, 23, 24].

In addition, another issue that should be clarified is the interaction modelling between CAAs and CSs and how retrieved CIs impact CAAs' behaviour and output [38, 40]. CS is the provider (the engine of CIs) which captures CI through a special transformation algorithm and carries it to such applications, as well as converting physical entities to virtual entities and inputting them into CAAs as parameters [30]. Requirements engineering and modelling for CAAs by the CA-UML is required in order to determine the required CI.

This chapter examines the challenges of depicting the behaviour activities of CAAs and proposes an extension notion in the form of the context-aware activity diagram to allow visualization of the flows of behaviour activities for CAAs and CSs. Furthermore, it adjusts the existing notations of the activity diagram and merges them with new notations of the context activity diagram to model overall behaviour activities of CAAs, and outlines the interactions for each swimlane and its meta-swimlanes [59]. More importantly for this classification, two modelling levels are suggested for the context-aware activity diagram notion, namely high-level interactions and low-level interactions.

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

Basically the activity diagrams explain the captured requirements for the system developers, which are usually specified by use case diagrams, and this chapter uses the new notion of the context-aware activity diagram to demonstrate the CAAs' functionalities to support the notion of the context-aware use case diagram to specify each context awareness requirement clearly through clear models to outline the adaptation actions and their behaviour. This thesis also uses context-aware use case diagrams first, and then moves to the next step in demonstrating CAAs' functionalities using a context-aware activity diagram notion, extending this thesis's approach of CA-UML to include an activity diagram to express CS behaviour, which is the external component that interacts with CAAs. More importantly, this chapter applies the requirements engineering aspects to visualize the CAAs' behaviour, which provides the developers with a clear understanding of the adaptation activities and how CIs affect the final actions of CAAs.

The suggested notion for the context-aware activity diagram of the CAA has three broad swimlanes as follows: User activity, CAA activity (controller) and CS activity. This chapter enables analysts to understand CAA behaviour in more depth, and also provides clear models and increases awareness of the differences between CS and CAA [32, 37].

It is necessary to extend the CA-UML approach to include classifications of CAA activities with new notations using an Activity Diagram, which needs a new swimlane for CS to model the interactions between the user, CAA and CS and to be understood more clearly. A further objective of the context-aware activity diagram is to clarify each CAA scenario by demonstrating each use case scenario (from the usage scenario template or use case diagram) into clear activities as workflows.

Unlike traditional systems, CAA requires a high level of analysis and modelling to achieve a high performance response without delays [33, 34]. This research investigates the extended notion of context-aware activity diagrams to model CAAs using an activity diagram to demonstrate the behaviour of CAAs.

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

4.2 Traditional Activity diagrams

Activity diagrams represent work-flows of actions modelled as activity nodes with connections by activity edges. Activity diagrams also provide clear flows of system activities in sequential steps which demonstrate the system details in low level modelling. In contrast to use case diagrams, the notations of Activity diagrams are clearer and do not include a set of relationships.

However, an activity diagram clearly expresses understanding about system behaviour as activities of the user, the system and their interactions. In addition, the activity diagram specifies the system objects, nodes, actions and their conditions which control the system services for the user [65, 99, 115].

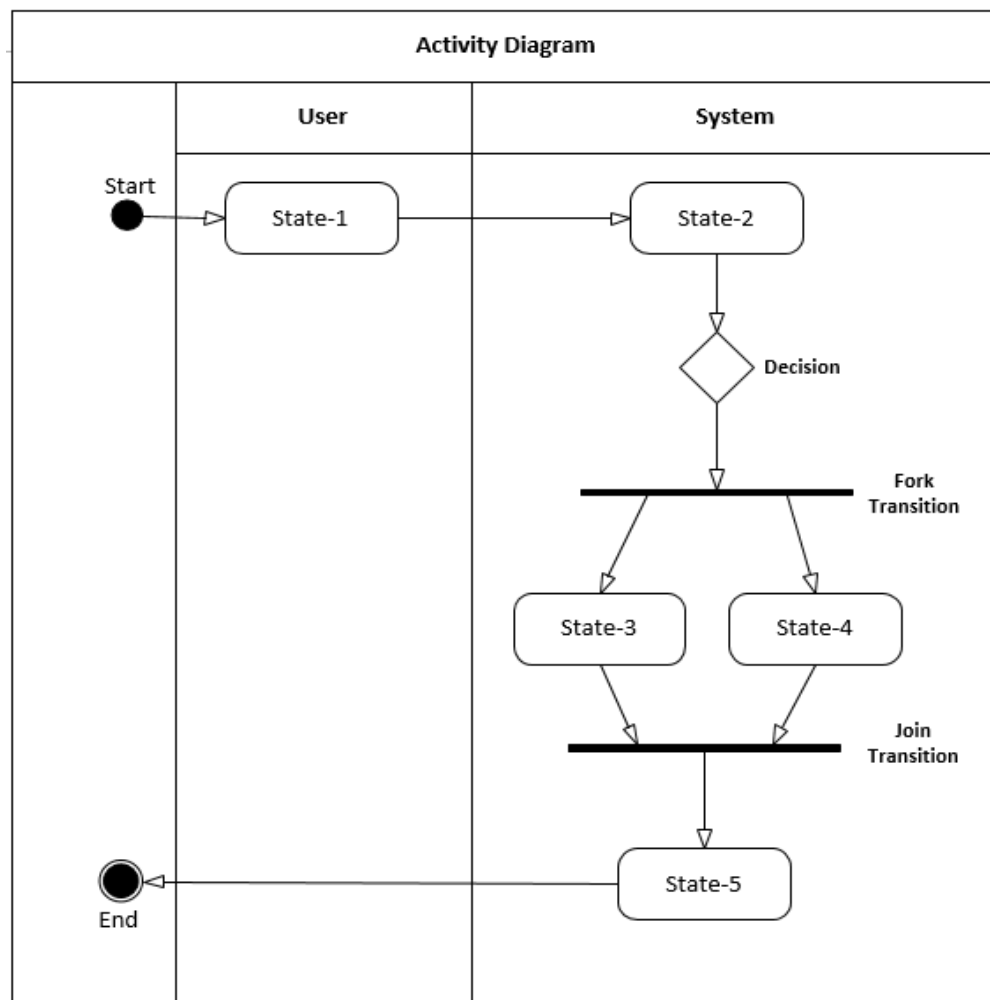


Figure 4.1: Activity diagram swimlanes and notations

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

Overall notations are depicted in Figure 4.1 of the activity diagram as follows:

- Swimlane

An activity diagram provides two swimlanes for user and system activities. Each swimlane classifies the behaviour of the system and external components that interact with system functions (such as the user); in addition, there may be more than two swimlanes on the activity diagram when different users interact with a set of systems. However, this swimlane is represented graphically as vertical solid lines on swimlane sides.

- Start node

The first step of the activities flows via the start node; usually the start node is out of the swimlanes of the user and the system. This node also links the user swimlane via an activity edge connection. This node is represented graphically as a bold circle in solid black.

- End node

The last step of the activities flows via the end node; the end node is also out of the swimlanes of the user and the system. This node links the system swimlane via an activity edge connection. This node is represented graphically as a filled circle in solid black inside a bigger unfilled circle.

- State notation

This notation has the ability to represent a set of activities for the system actions, executions, functions, operations and tasks, and these activities connect to each other via an activity edge to express the flows of activities within each swimlane of the user and the system. This notation is represented graphically as a capsule-shaped rounded rectangle.

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

- Decision notation

This notation creates multiple options for system actions depending on specific conditions. However, a decision notation has a single incoming activity (a single state notation connected with decision notation by an activity edge) and multiple outgoing activities (multiple state notations connected with a decision notation by an activity edge). This notation is represented graphically as a diamond shape.

- Transition

A controlling connection of flow activities which includes two types: a join transition to represent many incoming transitions and a single outgoing transition; or a fork transition to represent a single incoming transition and many outgoing transitions.

- Activity edge connection

A connection arrow specifies the direction of flow activities between the notations and nodes of the activity diagram. This connection is represented graphically as a solid directed line with an arrowhead.

4.3 CAAs Behaviour Modelling

Depiction of the dynamic behaviour of CAAs using an activity diagram is designated by their notations, which can visualize the flow activities of CAAs. However, the adjustable existing notations of the activity diagram provide awareness of CAAs' activities as well as expressing the context acquisition of CSs. In addition, it is suggested that a new concept of context activity diagram with new notations is necessary to enhance the existing notations: merging them together would provide clear steps of the adaptation activities and the constraints of CAAs to complete the behaviour activities. Table 4.1 compares the chapter concepts with the existing concept of the activity diagram.

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

Table 4.1: Comparison between traditional activity diagram, adjustable activity diagram and context activity diagram

Concept Name	System behaviour activities	CAAs behaviour activities	Self-adaptive activities
Traditional Activity diagram	√	×	×
Adjustable Activity diagram	√	√	×
Context Activity diagram	√	√	√

The following sections outline the two concepts to be merged for the proposed extension of context aware activity diagram as follows:

- Adjusting the notations of the Activity diagram for CAAs modelling
- Creating the notations of the Context Activity diagram for CAAs modelling

4.3.1 Adjusting the notations of the Activity diagram for CAAs Modelling

The basic notations can be adjusted for CAAs’ activities, making it possible to demonstrate the CAAs’ activities and operations and their conditions (as shown in Table 4.2). More importantly, this will allow the expression of CS activities and context acquisition, providing the data needed to execute CAAs’ processes and operations depending on conditions. In addition, the activity diagrams need more notations to outline the adaptation activities and context constraints, which affect each other depending on changes in the environment.

The use of the same notations that were previously used in the traditional activity diagram will make it possible to describe the works-flow between CAA activities and

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

CS activities. Furthermore, a new concept to include new notations may enhance the behaviour modelling of CAAs as well as specifying the change data captured by CS, which affect adaptation decisions and context constraints, which will help to control the end results for users in response to their needs and preferences.

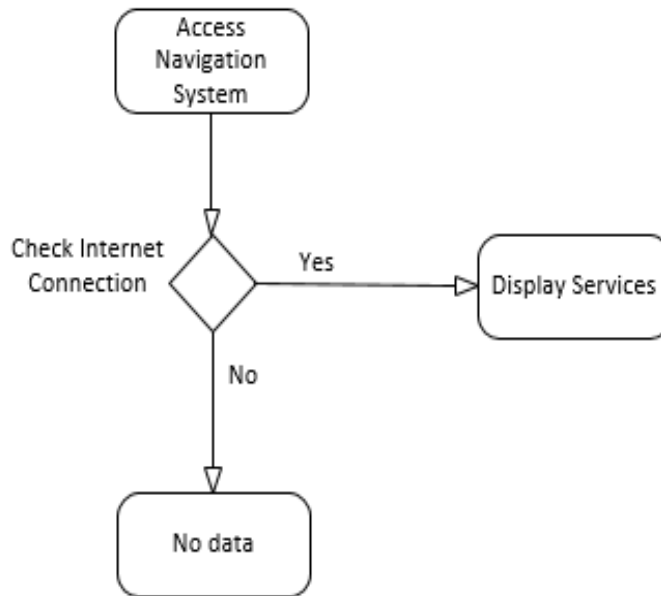
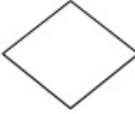





Figure 4.2: An example of adjustable notations of the activity diagram

The existing notations of the activity diagram still enable expression of the behaviour of CAAs' activities, as depicted in Figure 4.2, which requires the condition of an internet connection to display the services of the navigation system. Moreover, the state notations can represent the functions and operations of CAAs and the decision notation is also able to represent the conditions required to process the functions.

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

Table 4.2: Adjustable notations of activity diagram for CAAs

Notation	Traditional Usage	Adjustable Usage
 Decision	Specify a condition to make specific operations for system	Specify a condition to make specific operations for CAAs
 State	Represent the activities of system functions and operations	Represent the activities of CAAs operations and context acquisition
 Swimlane separation	Each swimlane classifies the behaviour of the system and the user	Each swimlane describes the activities within the corresponding category
 Activity Edge	System connection of flow activities	CAAs connection of flow activities

4.3.2 Creating the notations of Context Activity diagram for CAAs

Modelling

The suggested new concept of the context activity diagram including new notations enhances the existing notation of the activity diagram to enable it to present the dynamic behaviour of CAAs. However, by analogy to the activity diagram, this chapter proposes new notations of adaptation activities and context constraints to visualize the specific activities which impact the dynamic behaviour of CAAs' services as well as controlling the output actions for the user.

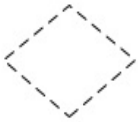




In addition, the behaviour of self-adaptive operations is the most challenging part of the modelling stage for system developers and these new notations are necessary to provide the effectiveness of activities and their constraints.

Furthermore, new notations are provided to depict adaptation decisions and their constraints, which are the real controlling activities of CAAs' behaviour which specify

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

the results of specific actions to fulfil the user's needs and outline the affected activities to produce specific services depending on the user's preferences.

Table 4.3: New notations of context activity diagram for CAAs

Notation	CAAs Usage
 Context Constraint	Represents CAAs' constraints in terms of checking CI values using special functions specified for adaptation actions
 Context Object	Stores the value of a CI
 Adaptation Action	Depicts adaptation action for CAAs' services
 Meta-swimlane separation	This notation is used to divide CAAs into three partitions: the user activity, the CAA activity, and the CS activity
 Context Activity Edge	CAAs' connection of flow activities of adaptation actions and context constraints

The above table (Table 4.3) of new notations for the context activity diagram is divided into three types depicting the flow activities of self-adaptive actions as follows:

- Context Constraint

This is a special notation for context constraints to specify the rule and adaptation conditions to make specific adaptation actions for the user as self-adaptive services. However, the context constraint notation has a single incoming adaptation action connected by a context activity edge (which may be a normal activity incoming) and multiple outgoing adaptation actions connected by a context activity edge (perhaps

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

outgoing to another context constraint). This notation is represented graphically as a dotted diamond shape.

- Context Object

A context object stores the value of a CI and is connected to activity states. 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 to which it is connected. This notation is represented graphically as a dashed rectangle.

- Adaptation Action

This notation visualizes the activities of adaptation actions for CAAs, which has the real task of changing the services for the end user depending on his/her personal preferences within the CAAs swimlane only. This notation is represented graphically as a dotted capsule-shaped rounded rectangle.

- Meta-swimlane

The meta-swimlane notation is used to partition a context activity diagram into three segments: the user activity segment, the CAA activity segment, and the CS activity segment.

- Context Activity Edge

This connection arrow uses the same element as the traditional activity diagram but for a special connection which specifies the direction of flow activities between the notations of the context activity diagram. This connection is represented graphically as a solid directed line with an arrowhead.

To support the concept of the context activity diagram, the example depicted in Figure 4.3 describes the way temperature affects the system's behaviour. Temperature is an example of CI as a parameter captured by CS and carried into an application that needs it to process a set of functions leading to the provision of self-adaptive services for users. In addition, the notations shown in Figure 4.3 express a practical usage of new

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

notations of the Context Activity diagram. Those notations describe the activities of adaptation actions and their constraints to make real actions as self-adaptive services.

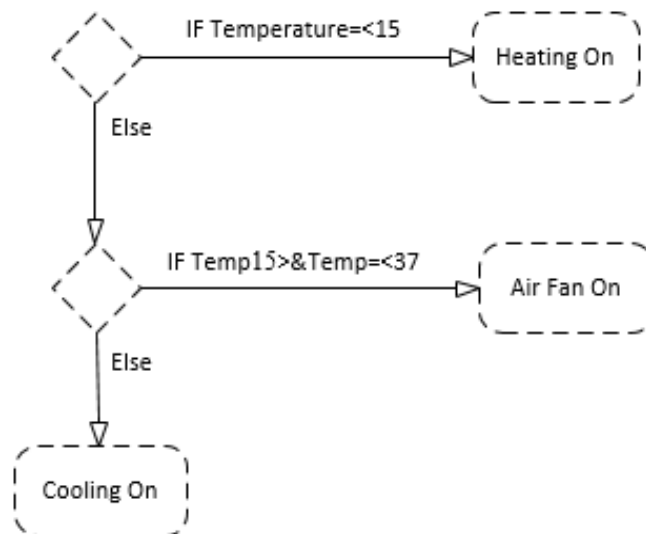


Figure 4.3: An example of the notations of the context activity diagram

4.4 A Context Aware Activity Diagram

The context-aware activity diagram notion proposed in this chapter uses both the adjustable activity diagram and the context activity diagram, merging the existing and new notations together to help to complete overall activities for the dynamic behaviour of CAAs and their components. This separation concept helps to achieve clarity of depiction between the activities of CAAs and CSs (which are depicted by the concept of the adjustable activity diagram notations) and the activities of adaptation actions of CAAs and their constraints (which are depicted by the concept of the context activity diagram).

In addition, CAA behaviour is affected by the CI which is retrieved from the CS swimlane and controlled by adaptation actions and their constraints (context constraints of pre-conditions or post-conditions) that manage the adaptation results in intended actions. Moreover, activity partitions (activity swimlanes) represent external entities or

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

attributes that carry out specific actions including the required values, and activity partitions that work as groups to carry out a series of actions that provide many details. Adaptation actions are required for the CAA to outline the functions, activity behaviour and CAA attributes.

Furthermore, CAAs possess many subsystems, and also other elements of software, hardware and electronics. CAAs need to realize all parts to provide services for the user with specific information at any time and place. Simulation of the dynamic behaviour of the CAA will define which part is important for the CAA, and also specify the purposes of any interaction with an external element and identify the different options of CAA responses.

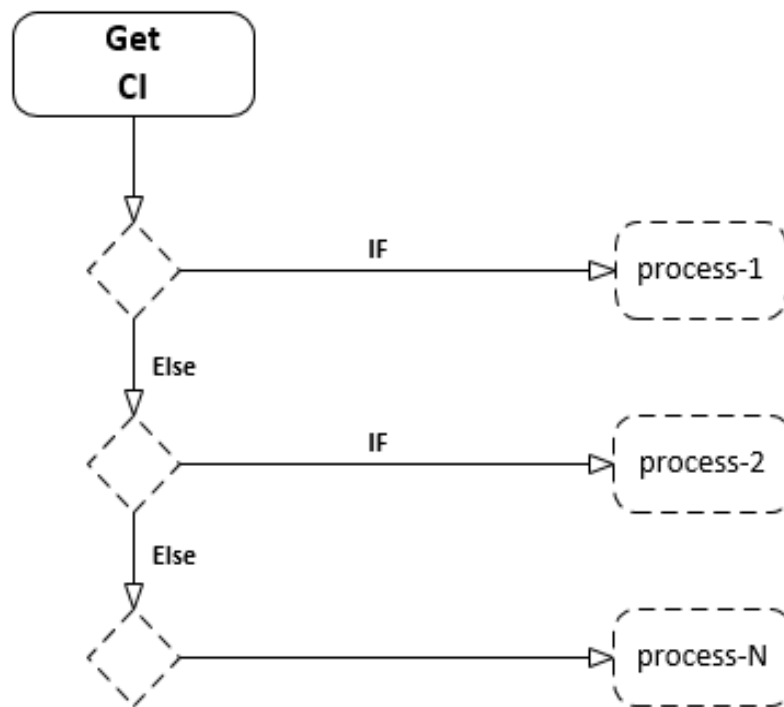


Figure 4.4.A: Context-aware activity diagram notations to show the activities of context constraint

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

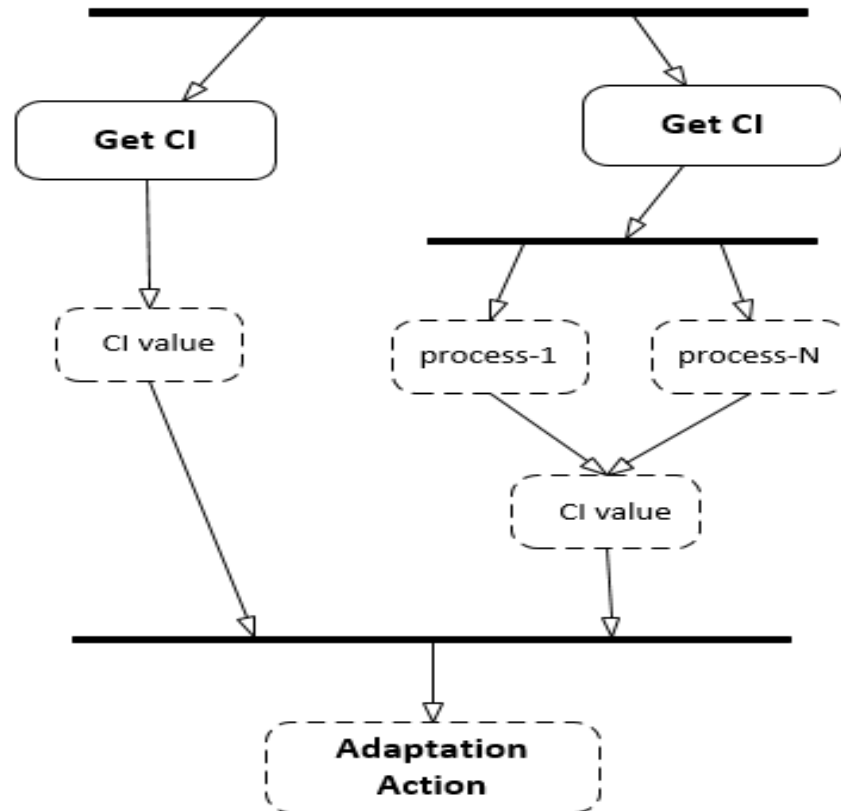





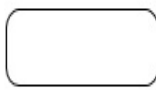




Figure 4.4.B: Context-aware activity diagram notations to show the activities of adaptation action

To support the context-aware activity diagram notion, Figures 4.4.A and 4.4.B express a set of notations which merge the existing notations of the activity diagram and the new notations of the context activity diagram to link the activities of CAAs and their adaptation activities. A context constraint is used by the IF-Statement function to specify specific processes to achieve the required self-adaptation actions. Furthermore, adaptation actions are used by different functions and operations to calculate the values of CI that produce a real action for the user.

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

Table 4.4: Notations for describing a context-aware activity diagram

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

4.4.1 Specifying context acquisition of CS behaviour using a context-aware activity diagram

The activities of CS context acquisition are depicted to determine the CIs which are captured by correct CS to specify suitable software services and ensure that data output is provided by the CS to identify the task queries and how they can manage and save changed CI.

CI is any physical, tangible object, and usually the main task of sensor devices (context services) is to capture specific CI and carry it to applications for different uses, such as location, vehicles, buildings, human beings, and devices. Delivery of the requirements of CAAs, which specify a set of methodologies, enhances the delivery of CI; the context acquisition of CS is required to deliver a specific value instance to CIs, which is

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

calculated to manage different services for users. This clarifies how CSs integrate with each other and how to send, receive and analyse CIs.

CSs' interactions may face challenges while capturing CIs, such as GPS, which tracks the user and updates his/her location, but requires different CSs. However, 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 that translates the physical entities into virtual entities, as shown in Figure 4.5.A.

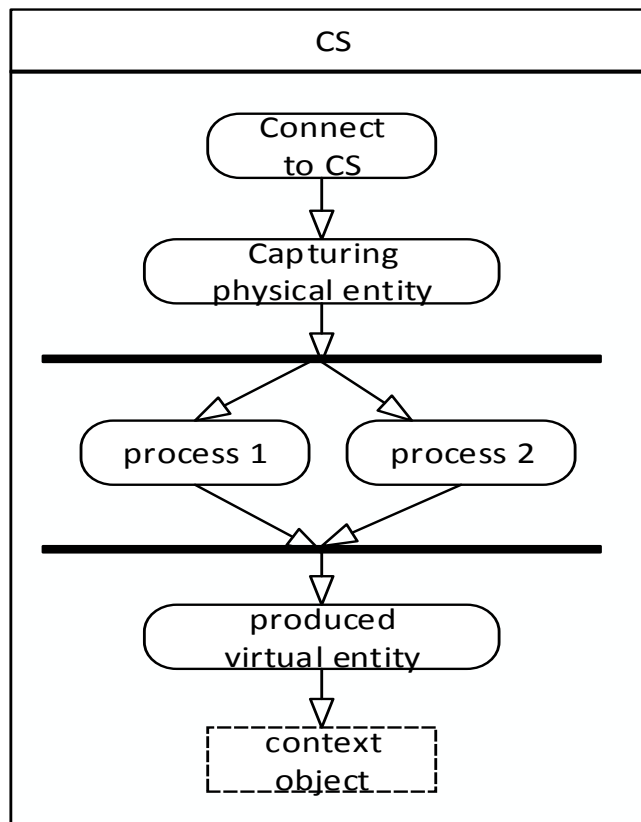


Figure 4.5.A: An illustration of CS activities

Furthermore, 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 CIs, such as temperature, speed, location, movement, photos, light and so on.

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

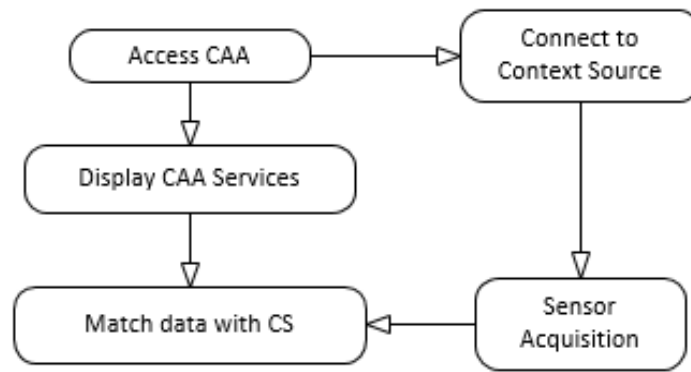


Figure 4.5.B: An illustration of flow-activities between CAA and CS

To support this, the diagram shown in Figure 4.5.B shows the practical usage of adjustable notations of the activity diagram and its uses. However, the existing notations make it possible to depict the dynamic behaviour of CAA and CS activities and their interactions.

4.4.2 Specifying the behaviour of Adaptation actions using a context-aware activity diagram

The context-aware activity diagram notion has combined the notations of two concepts to visualize all activities as models for the dynamic behaviour of CAAs which address important results as self-adaptive actions to allow the production of adaptation actions and context constraints. They also help to decrease the usage scenario conditions, because context constraints include conditions which make specific actions, although usage scenario conditions will be more detailed for each step.

In addition, the activity models of CAA are controlled by the requirements specified by the use case model with a streamlined flow and a clear identification and specification of all contexts and context information values, changes and domains, possibly affecting the CAA behaviour or adapting the service.

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

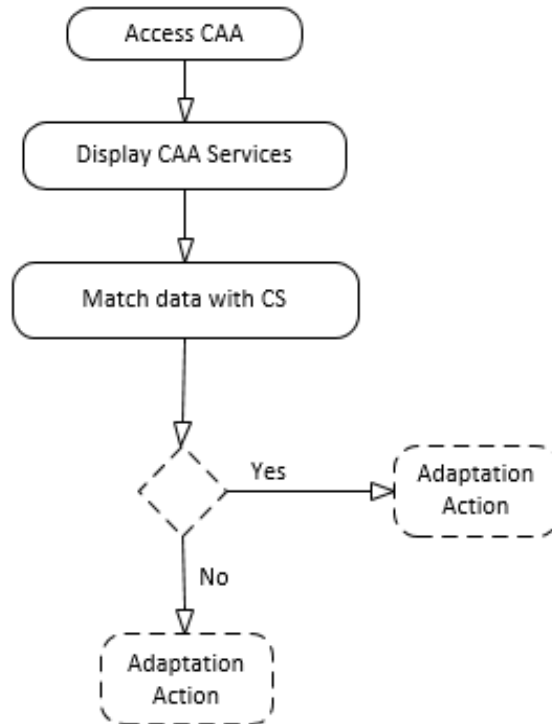


Figure 4.6: An illustration of adaptation actions and their constraints

The diagram shown in Figure 4.6 expresses the practical usage of the new notations of the context activity diagram and its uses. The existing notations provide the CAA operations, while the new notations provide the results of CAA operations as adaptation actions controlled by context constraints to depict the real actions that are able to change the system services of the CAA.

Simulating the real-time behaviour of CAAs is the most challenging part, especially when producing the output is affected by changes in CAA behaviour and user intentions, which are always changing and are detected by different CSs.

4.4.3 Standard Context Aware Activity diagrams

The context-aware activity diagram notion addresses the issues of adaptation and behaviour for CAAs' activities in responding to the changes of user using the new concept of context activity diagrams and their notations. However, this chapter extends the notations of the UML activity diagram and investigates CAA modelling challenges

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

by using the new concept of the context activity diagram. Context-aware activity diagrams also design adjustable levels of two-level modelling for CAA and extend the number of existing activity swimlanes from two to three. This serves to simplify complicated issues regarding CAA behaviour and interactions with other components step by step, by using current UML notations and new notations of CA-UML extensions.

In addition, the notion of activity diagrams proposed in this chapter shows the sequence and flow of actions and CIs activities performed by CAAs; it also illustrates how the activity model of CAA will be controlled by the requirements specified in the use case models with a simplified flow and a clear specification of all CIs and changes in values.

Furthermore, the main reasons to split the context-aware activity diagram notion into two levels of high-level interaction and low-level interaction are as follows:

- A standard modelling of high-level interactions outlines a general view of the main components (swimlanes) of CAAs, which illustrates the flow of activities between the user, CAA and CS;
- A standard modelling of low-level interactions provides more detail within each component interaction (meta-swimlane) of the user, CAA, or CS, which demonstrates the interaction activities of each swimlane;
- Both standard modelling forms provide clear visualization of CAAs' components and their interactions;
- The high-level interactions graphically represent the swimlane as double vertical solid lines on the swimlane sides to understand the activities of CAAs' components;
- The low-level interactions graphically represent the meta-swimlane as vertical solid lines on both sides to understand each component's activities.

In other words, the context-aware activity diagram notion (as depicted in Figures 4.7.A and 4.7.B to manage the interactions of CAA components) uses double vertical lines in high level interactions to classify the main parts of CAAs: user, CAA and CS.

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

Figure 4.7.A expresses graphically a set of classifications: the first classification is for the user who interacts with the CAAs' facilities; the second is for the CAA which provides the adaptation services for the user, and the third is for the CS which captures CI around the user and carries it to the CAA.

Each classification is divided into many meta-swimlanes, as modelled in Figure 4.7.B, to show the interaction activities of each CAA component.

Graphically, Figures 4.7.A and 4.7.B classify the dynamic modelling of the CAA into three broad swimlanes of user, CAA and CS activities as follows:

- User Activity

The user 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 the CAA 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 later use. The security requirement is a non-functional requirement of such a system, and in CAA modelling, the security activities are within the user swimlane, which means that security requirements are outside the scope of the CAA.

- CAA Activity

The objective of the context-aware activity diagram notion is to model the dynamic behaviour of the CAA, which needs an extra swimlane for the CS as an external component; the difference between traditional systems and CAAs is that CS is responsible for feeding the CAA via different CIs and collecting the parameters needed for the CAA by itself, in contrast to parameters in traditional systems, which are stored in a database or inserted by the user.

Furthermore, the CAA can retrieve a set of data from CS databases by interacting with the CS server and updating the user with a variety of CIs when required, such as changes in the running of CAA depending on user environmental conditions.

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

Moreover, the CAA provides the user with adaptation actions, which are controlled by specific constraints to respond to the user's needs and preferences. Adaptation actions also respond to the user as self-adaptive actions to monitor the changes to the surrounding environment.

- CS Activity

The main objective of the CS is to read the CI, which senses the required object. In addition, there are numerous processes relating to acquisition senses, including starting the acquisition of sensors and reading the different objects required to create CIs to be sent to the CS Server and saved in the CS databases.

A model of related activities describing acquisition senses by the activity diagram is used to clarify and specify the full range of CAA behaviour for designers, the acquisition senses being the most complex part of CAA modelling.

Acquisition senses through CS provide the CAA with the required information. Furthermore, model CS activities are important for several reasons: they enable CAA designers to know which CS will interact with CAA, understand CI parameters and know the CS database schema.

The following section investigates an example of the proposed two levels of modelling and when each one should be used.

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

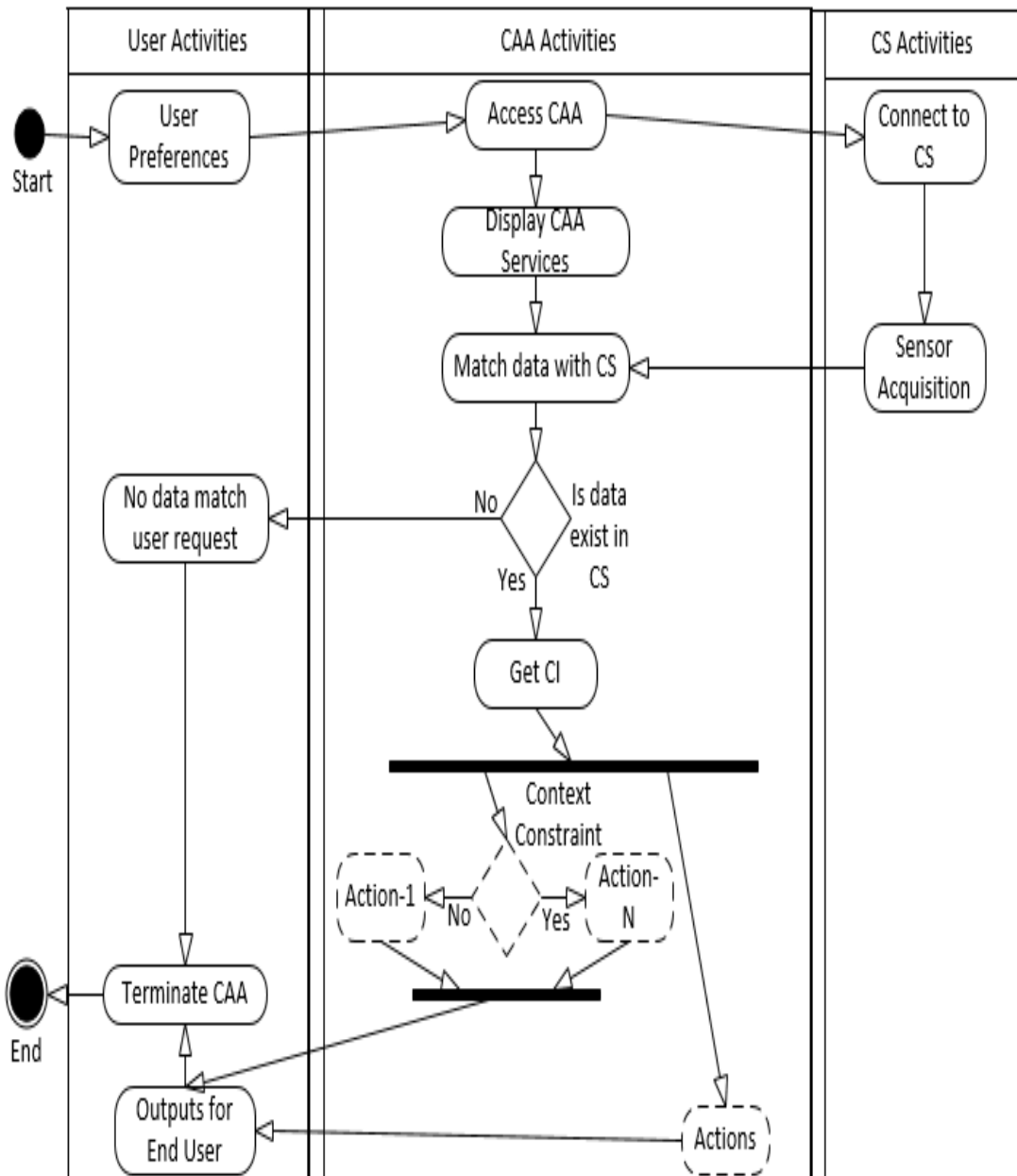


Figure 4.7.A: Context-aware activity diagram modelling of high-level interactions

A high-level mapping of the CAA swimlanes' interactions is depicted in Figure 4.7.A. This standard modelling illustrates an abstract modelling of CAA behaviour as well as using the overall notations of the context-aware activity diagram notion. In addition, Figure 4.7.B represents another standard modelling for a low-level mapping of each swimlane interaction, which helps to clarify the integration issues between the swimlanes.

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

This standard modelling demonstrates an expanded modelling of CAA components' interactions, also using the merged notations of both concepts of the adjustable activity diagram and the context activity diagram.

In Figures 4.7.A and 4.7.B, we can observe in the high-level interactions a new swimlane for CS which needs a particular kind of classification: the swimlanes of interaction activities express the main components of CAA. This chapter outlines three swimlanes, namely the user profile swimlane, the CAA swimlane and the CS swimlane. This ensures that CAA activity models are clear and easily understood in order for designers to translate these models to the source code and scripting level.

Figure 4.7.A illustrates the tracing activities for the main components without describing other details for each component. Figure 4.7.B demonstrates more sub-classifications of interaction activities for each component, as depicted in the standard modelling of low-level interactions.

For clarity, interactions' classifications are given a unique name for each swimlane for the user, Context-Aware Application and CS and their meta-swimlanes as follows:

Users' sub-swim-lanes: (User-1, User-2 User-N)

CAAs' sub-swim-lanes: (Sub-CAA-1, Sub-CAA-2 Sub-CAA M)

CSs' sub-swim-lanes: (Sub-CS-1, Sub-CS-2 Sub-CS-K).

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

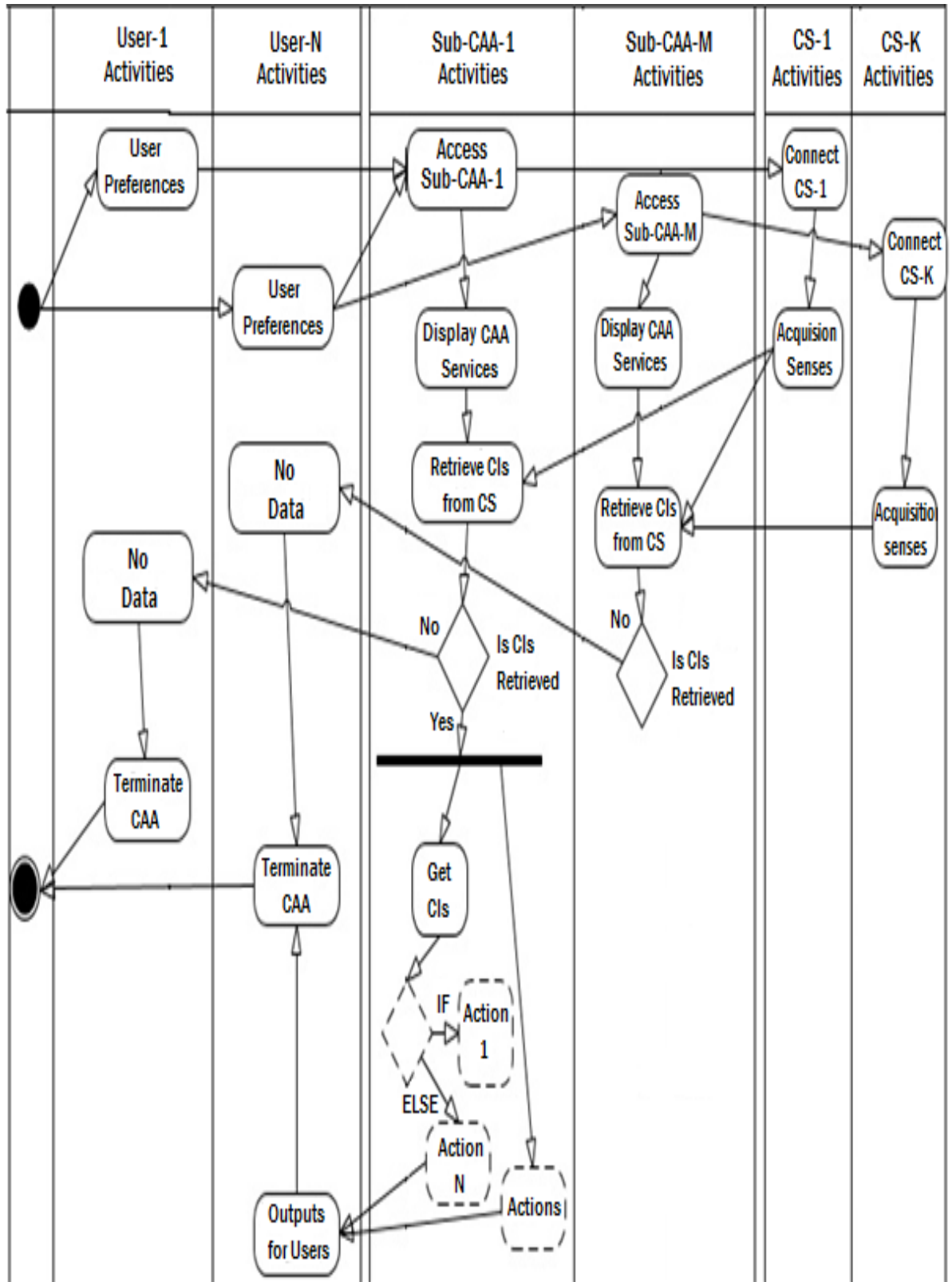


Figure 4.7.B: Context-aware activity diagram modelling of low-level interactions

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

4.4.4 A practical example of a Temperature Control System using a context-aware activity diagram

Consider an example to support the context-aware activity diagram notion which applies high-level interactions to model the dynamic behaviour of CAAs using adjusted and extended notations to demonstrate their effectiveness. An example of a temperature-control system is shown in the following diagram, which depicts the activities of temperature-control system and CS and outlines the adaptation actions and their constraints required to implement temperature-control system services [30]. This modelling of temperature-control system is an example of a CAA that could be run by mobile devices to respond to user preferences and needs.

This example shows different issues for CSs interacting with CAAs using the context-aware activity diagram notion to address the activities modelling for CAAs. Moreover, this example uses high-level interactions to express the CAA activities of temperature-control system and also requires abstract modelling via the standard modelling of high-level interactions, which includes a single system to depict the behaviour of temperature-control system in different activities. Acquisition and input of many variables of Atomic CIs relating to each other lead to the production of CIs as services (adaptation actions); the development of adaptation results in the end action, such as temperature-control system.

Temperature-control system services are important for many reasons: to ensure users' comfort, to save energy and to act as self-controllers in cases when temperature-control system 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.

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). Typically, any system needs to go through a series of different stages before implementation to ensure that it is

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

accurate and is meeting the correct objective, for which the user requirements must be specified carefully.

The user requirements in this practical example are to use a self-adapting temperature-control system 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 example (depicted in Figure 4.8 and summarised in Table 4.5) the temperature range classification depends on user preferences, which are defined as follows:

- Temperature-control system applies Heating mode when the temperature of the user's room is less than or equal to 15 degrees.
- Temperature-control system applies Warm mode when the temperature of the user's room is more than 15 degrees but less than or equal to 25 degrees.
- Temperature-control system 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.
- Temperature-control system applies Cooling mode when the temperature of the user's room is more than 37 degrees.

The temperature-control system inputs a range of temperatures to carry out the required actions: Output actions depend on the temperature range, which is controlled by a special function (on/off, PID). In Figure 4.8, describing the Temperature range, there are four output actions (heating, warm, air-fan and cooling).

Table 4.5: Summary of temperature-control system behaviour activities by context-aware activity diagram

Temperature-control system	Adam
CAA Adaptation Action	Heating, warm, air-fan and cooling depending on temperature
CS Acquisition Sense	Temperature sensor produces surrounding temperature for the user
CI Type	Atomic CI (Temperature)

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

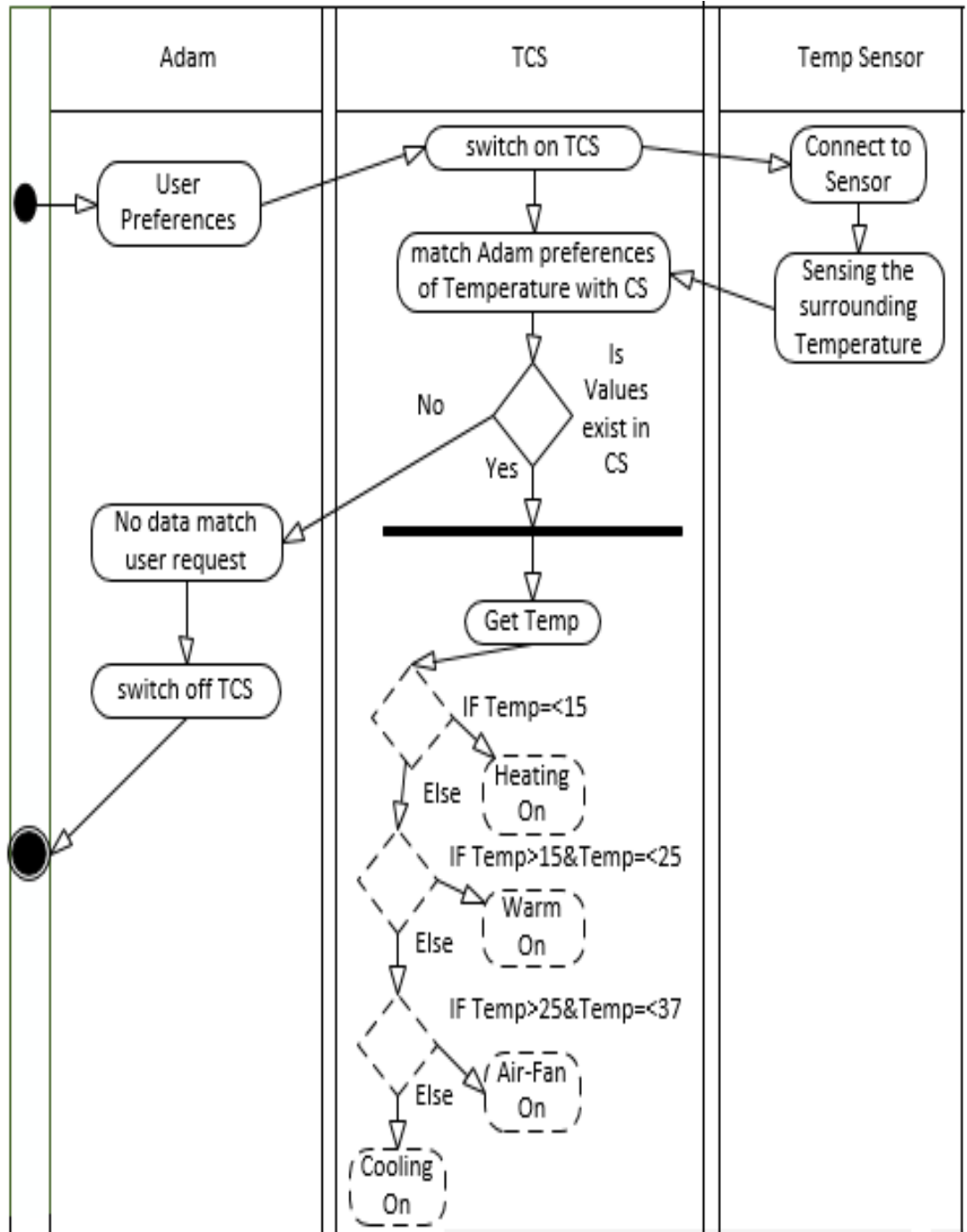


Figure 4.8: Context-aware activity diagram activities of temperature-control system using high-level interactions

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

4.5 Summary

This chapter has extended the UML activity diagram by adjusting its notations to describe the dynamic behaviour of CAA and CS activities, and has also created a new concept of the context activity diagram including new notations to address the adaptation actions and their constraints with regard to CAA and its services. The notations of these two concepts are merged to demonstrate the flow of activities of CAAs' behaviour.

Context-aware activity diagrams provide adjustable levels of two types of standard modelling for CAAs' behaviour, identifying a new swimlane for CS activities to include the name of CS and their context acquisition. CAAs' classifications of the user, CAA and CS also provide the ability to understand the behaviour of CAAs and their components and visualize the interaction modelling between CAA and CS and how retrieved CI impacts CAA behaviour and services. Also, each swimlane for each component has interactions with its parts as meta-swimlanes: in other words, the context-aware activity diagram includes a suggested swimlane for CS, and has proposed new notations which are able to depict all CAAs' components as activities and their adaptation actions and constraints.

More importantly to this chapter, the context-aware activity diagram notion involves merging a set of subsystems working as one system using extension notations of the activity diagram, which enables the separation of concerns between the functions, operations, adaptation actions and constraints of CAA to be performed in reaction to changes in the CAA environment.

These new notations and swimlanes in the context-aware activity diagram notion should be set out in more detail in the Usage Scenario (the usage scenario usually begins when the use case diagram is created to provide more details using specific steps as a template).

Based on the usage scenario of each use case, the activity model is depicted to represent the flow of actions, activities or processes that should be performed by the CAA; it

CHAPTER 4. AN EXTENSION OF ACTIVITY DIAGRAM TO MODEL THE BEHAVIOUR OF CONTEXT AWARE APPLICATIONS

handles more low-level details of system functionalities while demonstrating the possible behavioural actions or responses of the CAA.

Furthermore, the CAA investigates the highest level systems and challenging factors in the CAA, such as adapting to changes in CI. In terms of context, activity diagrams can be described in terms of granularity and used to demonstrate situations at different levels for CAA behaviour. The final development output should provide an implementation of each aspect relating to the activity model, processes, decisions, and synchronized processing.

Moreover, this chapter has provided an example of a temperature-control system to illustrate the merging of the two concepts for the context-aware activity diagram notion, expressing a practical usage of an adjustable level for the standard modelling of high-level interactions (the standard modelling of low-level interactions will be demonstrated in the case studies in Chapters 6 and 7 with a set of practical usage diagrams) including the existing notations and new notations for the dynamic behaviour of CAA modelling.

In the following chapter, this thesis suggests a new notion, namely the context aware class diagram, to model the structure of context aware applications.

5. An Extension of the Class Diagram to Model the Structure of Context Aware Applications

Objectives:

- Extend the UML class diagram to depict the structure of CAAs and their functions;
- Suggest a new concept, the context class diagram, for the modelling and design of CAAs;
- Suggest a new shape to express the static parts of contexts;
- Design a standard modelling for CAAs' structure;
- Specify the CAAs' components in terms of class shapes and context shapes using the new notion of the Context Aware Class Diagram;
- Use the context-aware class diagram to design each object for CAAs and their functions, as illustrated by the context-aware use case diagram as CAAs' functionalities and demonstrated by the context-aware activity diagram as CAAs' activities in the previous chapters.

5.1 Introduction

The class diagram is a standard modelling design to build the static aspects of traditional systems. Class diagrams also map the system classes and connect them via different relationships using an object-oriented model as a sequence of classes. In other words, the class diagram is the backbone for the design of any system's structure and components. The class diagram possesses the main characteristics of system classes and their relationships, which describe the system objects, attributes and methods. The main shape of class diagram elements is the class shape, which is divided into three compartments: the class name, attributes and operations. Class diagrams also represent the system components and objects as hierarchy classes.

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

In the design stage, the developers use the class diagram and the entities' relationship diagram to identify the system objects and their parameters before the construction stage; also, the class diagram supports the design stage and the system developers usually start using the class diagram when the modelling stage is completed (the use case diagram and the activity diagram are parts of the modelling stage). However, this thesis provides a clear approach by using an extension of the use case diagram and the activity diagram for the modelling level and an extension of the class diagram for the design level: these clarify all related objects and their contexts for CAAs, which are needed to execute the main functions specified in the previous chapters. More importantly, this chapter applies the requirements engineering aspects to design the context objects which provide the developers with a clear understanding of the CAAs' structure before the construction level.

Typically, the design stage of software applications 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. In the meantime, the class diagram is important in the design stage, as it provides information and connections between attributes 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 [118].

Typically, the UML class diagram has the powerful design concepts to model all different types of information systems but still needs an extension approach to design the contexts' aspects to model every CAA component for clarity of modelling and design. This chapter investigates the design stage of CAAs and suggests a new extension notion to enhance the class diagram for CAAs' structure modelling; the new notion is called the *context aware class diagram*.

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

A more important issue is that CAAs' structure modelling by the UML class diagram is applicable and the classes' models have the ability to design all object types for CAAs, which are the most popular and widely used approach for set types of software applications. Class diagrams apply the concept of the object-oriented model to express the components' structure for CAAs, but this structure requires evaluation to represent context awareness requirements as objects and their properties and functions. Furthermore, class diagrams are necessary to outline CAAs' modelling and complete their life-cycle development for clarity and ease of coding and implementation.

The importance of the class diagram is that it provides a low-level description of any system and prepares the system for the final models and the coding phase, while the objective of the context-aware class diagram notion is to simplify CAAs' development processes. In addition, depicting CIs with classes in this notion is helpful to specify the related information on contexts for the CAAs' design phase [94, 101, 119].

Class diagrams for ubiquitous computing show the proliferation of CAAs' connection with applications, and their structure components also specify the information for the properties of people and devices. Hence, the challenge for UML diagrams when modelling CAAs is the integrated applications acting in different environments and situations around the user.

A further issue involves sensors and how improving the capturing tools for CIs and their changes affects CAAs' performance. CAAs' objectives are to be aware of the user's personal environment, preferences, behaviour and needs and to customize CAAs' requirements to respond to user expectations.

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

5.2 Traditional class diagrams

Class diagram elements possess a main element of class shape, divided into three parts, and relationships to link a set of classes, with special constraints also providing a special meaning for each relationship. Hence, this section describes each part of the main shape of the class and specifies each relationship's meaning and its practical usage. The class diagram uses the visibility markers for each attribute (as depicted in Figure 5.1), which control authentication access, to specify who is able to access the information of the class [111].

Class attributes provide five visibility markers: the visibility of private attributes is used only inside the class and hides its information from other classes; the visibility of public attributes is used to allow other classes to access and retrieve its information; the visibility of protected attributes is used for inheritance, retrieving from the parent class to the child class, as well as allowing the use of the same properties and methods; the visibility of derived attributes is used to compute values from different attributes without storing; the visibility of package attributes is used to allow other classes to be visible within the same package.

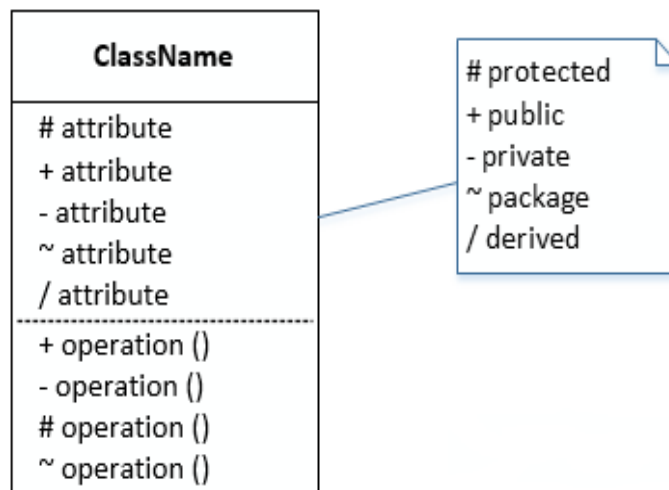


Figure 5.1: The partitions of class shape

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

The main elements of the class diagram are class shape and its relationships, as follows:

- Class shape

Class shape is graphically represented as a hollow diamond shape divided into three parts and classes are usually designed as a hierarchy model. However, class shape is divided into three partitions as follows:

- Class name: the first partition is the header for the class name and is usually written in bolded capital letters in the centre.
 - Attributes (properties): the second partition is the middle partition, for attributes to list all objects related to the system as classes and specify their properties (such as int, float, bool, string and so on).
 - Operations (functions): the third partition is the bottom partition, to list the class operations to execute specific functions for the system (such as set or get functions).
- Relationships

There are different types of class relationship, and each one indicates a special meaning as follows:

- Generalization

A generalization relationship acts for inheritance between classes to get parent properties and methods to child classes. This relationship is also divided into three lines for special meanings: a solid line with a black arrow for class, a dashed line with a white arrow for the interface class and a solid line with a white arrow for the abstract class.

- Association

An association relationship connects the structure for a set of classes and its arrow indicates the relationship type by specifying the class direction. This general relationship is graphically represented as a solid line.

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

Association gains different types of relationship, such as multiplicity, composition, aggregation and dependency, and the navigability of the line arrow identifies the relationship types.

The most common relationship type is multiplicity, which specifies how many objects are used between different classes (objects). This relationship provides different types of relationship, such as one-to-one relationships and one-to-many relationships, and outlines symbols as numbers and star symbols as follows:

- * Symbol means 0, 1, or more classes
- 1 Symbol means 1 class only
- 2..4 Symbol means classes 2 and 4 between
- 3.. Symbol means 3 classes or more

Another relationship type is composition, which refers to the relationship between parent and child if the child is not independent and the child class is part of another class, such as a room or part of a house. This relationship is represented graphically as a solid line with a black diamond.

Another type of association relationship is called aggregation, which applies the same meaning as composition for the relationship between parent and child except that the child is independent and may be part of the parent class or other classes, such as the student class being part of the father class. This relationship is represented graphically as a solid line with a white diamond. The final type of association is the dependency relationship, which is used temporarily if a class depends on another class. This relationship is represented graphically as a dotted line with an open arrow.

5.3 CAAs Structure Modelling

The class diagram is a powerful design concept for an object-oriented model but is still limited to the design of all components of CAAs and their changeable environments. However, adjustable existing elements of the class diagram can express the basic context objects (classes) of CAAs but still need a new shape in order to visualize the

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

context entities (contexts) to fulfil the CAA's modelling needs using an extension approach to design overall components of the structure of CAAs and their contexts.

The structure components of CAAs can be depicted using a class diagram to provide clear elements that enable the creation of object-oriented models of such a system. The elements can also specify the CAAs' classes and their relationships, although CAAs' structure modelling needs a new shape to express the CAAs' entities as contexts, properties and functions.

The following sections provide two concepts including new elements to design the context aspects that are required for CAAs' objects, using existing elements and their relationships of the class diagram together with new elements to express contexts and their properties. A new relationship is suggested to link the extended shapes with existing classes. In addition, the context-aware class diagram suggests a notion that merges elements of the two concepts to express the structure components of CAAs and their contexts.

Table 5.1: Comparison between traditional class diagram, adjustable class diagram and context class diagram

Concept Name	System structure	CAAs structure	Context structure
Traditional Class diagram	√	×	×
Adjustable Class diagram	√	√	×
Context Class diagram	√	√	√

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

The following sections outline two concepts to be merged in this notion for the proposed extension of the context aware class diagram as follows:

- Adjusting the elements of the Class diagram for CAA modelling
- Creating the elements of the Context Class diagram for CAA modelling

5.3.1 Adjusting the elements of the Class diagram for CAA modelling

CAAs' structure can be mapped by adjusting the elements of the class diagram, which is widely used to represent CAAs' objects, properties and operations (such as the stereotypes approach). However, class diagrams were developed for set types of software applications and still require an extension approach to fulfil the context awareness requirements of CAAs' structure. They can also be extended to express the context entities (instance values of contexts) using a new shape which always affects CAAs' services.

To support this concept of the adjustable class diagram, the diagram shown in Figure 5.2 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 in Figure 5.2. 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. However, render engine class is used as a render engine for all maps needed for the navigation system.

This class of render engine depends on data gathered from other classes, such as the map element, city and road. Moreover, the class of render engine does not include instance values of CI. Figure 5.2 adjusts the basic class usage to represent CAAs' classes, such as the class of the render engine, which can show the main properties and operations of the navigation application to be context aware of the user's location. Typically, CAAs' classes are still unable to represent context entities' details, such as context properties.

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

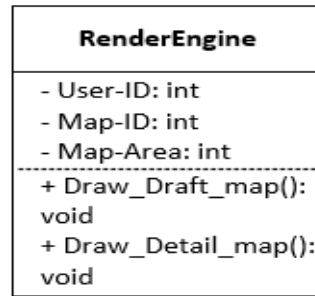
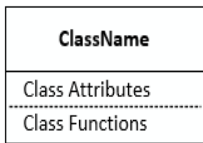







Figure 5.2: An example of the adjustable elements of the class diagram

In other words, the existing shape of the class enables context objects to be represented. However, to complete the structure of CAAs, a new shape is required to represent the contexts and their properties and functions, which are captured and transformed from a physical to a virtual state by the CS.

Table 5.2: Adjustable elements of class diagram for CAAs

Element	Traditional Usage	Adjustable Usage
 Class	Shape presenting system objects divided into three parts: class name, attributes and operations	Shape presenting CAAs' objects divided into three parts: class name, attributes and operations
 Generalization	System relationship to inherit properties of parent class to child class	System relationship to inherit properties of parent class to child class
 Association	A general relationship to connect a set of classes for system structure	A general relationship to connect a set of classes for CAAs' structure
 Aggregation	An independent relationship between system classes of parent and child	An independent relationship between CAAs' classes of parent and child
 Composition	Not independent relationship between system classes of parent and child	Not independent relationship between CAAs' classes of parent and child
 Dependency	System relationship between classes depends on another class	CAAs' relationship between classes depends on another class

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

5.3.2 Creating the elements of the Context Class diagram for CAAs modelling

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 CAAs.

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 CAAs' databases, which are always changing. Furthermore, context values are always changing, which affects the system results. In addition, the difference between the design stage of the system and CAA is that the parameters of the system may be stored in databases or inserted by the user through system run-time, but in CAAs 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 CAAs' databases (the last output of CAAs may be stored in the database without awareness of CI changes).

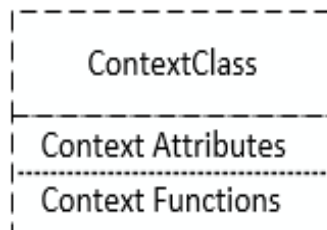


Figure 5.3: An illustration of shape partitions for the context class diagram

To support this concept of the context class diagram representing the context properties and functions, a new shape is used (as depicted in Figure 5.3), graphically represented as a hollow diamond dotted shape divided into three parts.

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

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 centre. A name is specified for each context to simplify the design stage of CAAs' structure, which outlines entities affecting CAAs' 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 CAAs. Examples of context attributes include int, float, bool, string and so on.
- Context functions: the third partition is at the bottom of the context shape and lists the functions calculating the values of CI to fulfil the needs of CAAs' services, such as retrieving the value of traffic status or the value of remaining time.

5.4 A Context-Aware Class Diagram

The context aware class diagram notion 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 CAAs.

Another important aspect of the context class diagram is that it uses the special relationship of *utilization*, which allows CAAs' classes to interact with contexts to retrieve the values of CI. Continuous retrieval is the solution to capture the context values in the user environment, which are constantly changing. Furthermore, this notion outlines the difference between class functions and context functions: class functions are executed by CAAs, while context functions are executed by CSs.

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

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

- What must the CS do for context awareness objects?
- How will the CAA fulfil the context awareness requirements?
- What classes will need to implement CAAs 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?

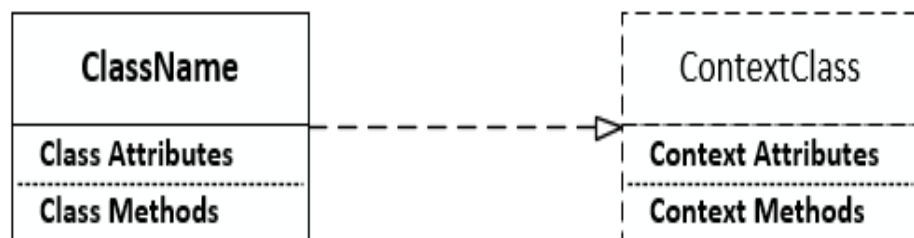


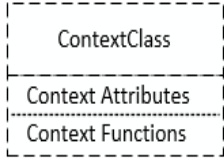
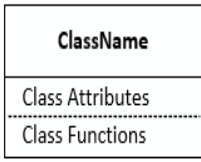






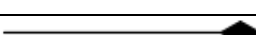
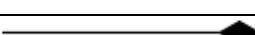


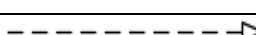
Figure 5.4: An illustration of elements for the context aware class diagram

In Figure 5.4, we can observe that the shapes of both class and context are merged for the context-aware class diagram notion and linked using a new relationship – the *utilization* relationship – which fulfils the context awareness objects by retrieving the values of CI.

In a simple definition of the context-aware class diagram notion, the concept of the class diagram creates a summary of all objects for CAAs and the concept of the context class diagram creates a summary of all entities for CAAs.

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

Table 5.3: Elements for describing a context-aware class diagram

Context-aware class diagram notations	
Context related notations	UML notations
	
Context	Class
	
Generalization	Generalization
	
Association	Association
	
Aggregation	Aggregation
	
Composition	Composition
	
Dependency	Dependency
	
Utilization	

5.4.1 Specifying the structure classes of CAAs' components using the context-aware class diagram

The basic classes have the ability to specify the objects for structure components to design CAAs. Traditional systems are well designed by class diagrams to express the system functions to store different services and the necessary parameters for normal system databases for multiple uses; conversely, the situation in CAAs cannot be completed without context values, which are collected by different CSs to represent the context awareness requirements to affect the CAAs' outputs.

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

However, using existing elements of the class diagram, it is possible to model the CAAs' components and their functions and properties by adjusting the usage of class diagrams to model the static structure of CAAs.

In addition, designing the context awareness objects as classes helps to apply CAAs' functions. 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 fulfil the context awareness requirements of CAAs.

Furthermore, in the context-aware class diagram, the existing classes' usage is adjusted to design the users' preferences and set their properties, and the context awareness objects can be structured by class diagrams. The following section will extend the elements of the class diagram to complete the design of the CAAs' structure. In other words, the component structure of CAAs can be designed using the elements of the class diagram to design the user and system components as classes.

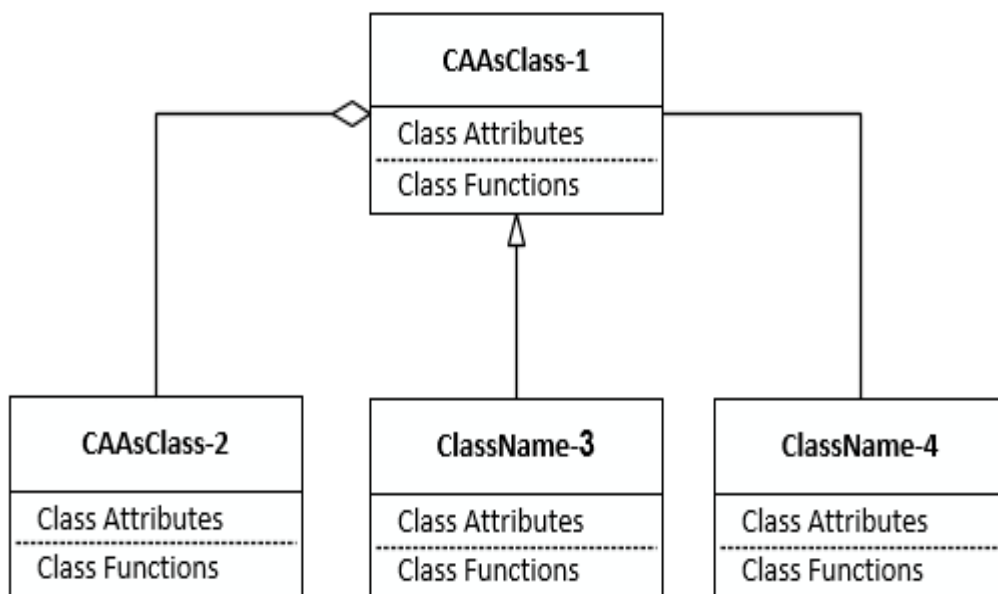


Figure 5.5: The structure classes of CAAs' components

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

5.4.2 Specifying the structure contexts of CAAs components using a context-aware class diagram

The context-aware class diagram notion 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: CAAs' classes are designed to retrieve the CIs' values from CSs. This approach provides solutions for CAA designers, who should focus on the availability parameters produced by CSs to fulfil CAAs' requirements of context awareness and user needs.

In addition, context shapes are designed in the context-aware class diagram notion to outline the exact CI values needed for CAAs' structure design. CS creates the physical entities as contexts to make the CAAs' 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 notion expresses the context values in the design stage of CAAs to identify the parameters that should be retrieved for classes of CAAs. The shapes of both classes and contexts will decrease the complexity of the structure of CAAs 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).

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

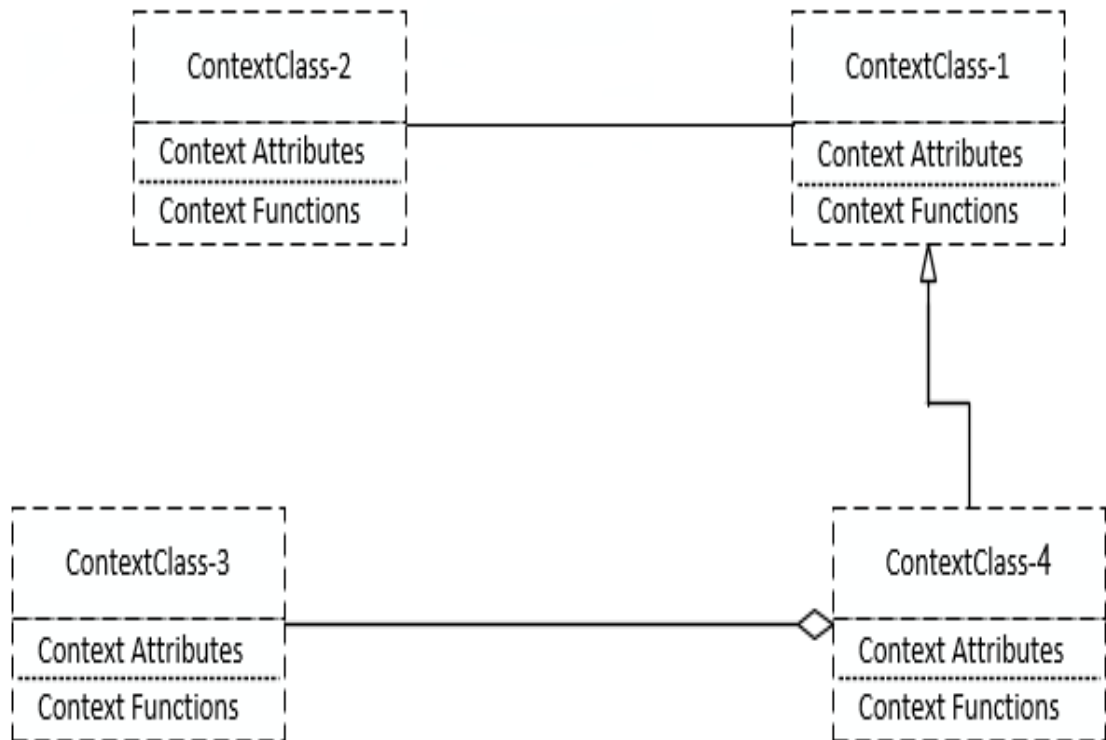


Figure 5.6: The structure contexts of CAAs' components

5.4.3 A standard Context Aware Class diagram

The extension notion of the context-aware class diagram provides a standard modelling for CAAs' structure design which expresses the elements of both the adjustable class diagram and the context class diagram and their relationships to design the context awareness requirements and the structure components of CAAs (as designed in Figure 5.7).

The modelling of the context-aware class diagram notion specifies the attributes and functions for classes and contexts. Classes' shapes outline the properties and operations of CAAs' objects, while contexts' shapes outline the properties and operations of CSs' entities. The entities are produced by CSs to sense specific physical entities and to be carried to such applications, while the captured entities are transformed into virtual form.

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

These virtual entities are the contexts that are important to complete the processes of CAAs. Contexts are changes and their data cannot be stored in the databases of CAAs. The CIs' values will be used in the run-time of CAAs.

In addition, the designers of CAAs can specify the specific values of context-awareness requirements at this stage and which CIs should be retrieved. In Chapters 6 and 7, case studies of navigation system and weather forecast system are provided to support theoretical studies of the context-aware class diagram and their elements using the CA-UML approach.

Furthermore, the standard modelling for this notion depicts the merged elements of the two concepts, which are connected by the new relationship of *Utilization* to fulfil CAAs' classes with the required CI values for different purposes. In Figure 5.7, we can observe two different types of shape: one for classes and another for contexts. However, the users of CAAs cannot specify which CIs should be retrieved to achieve the self-adaptive services. The designers of CAAs have to provide many options for users in different situations and circumstances that allow the users to achieve the adaptation outputs without any complexity. The following chapter, which presents a real-life case study of a navigation system, will investigate the context-aware class diagram notion for the appropriate modelling and design of CAAs.

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

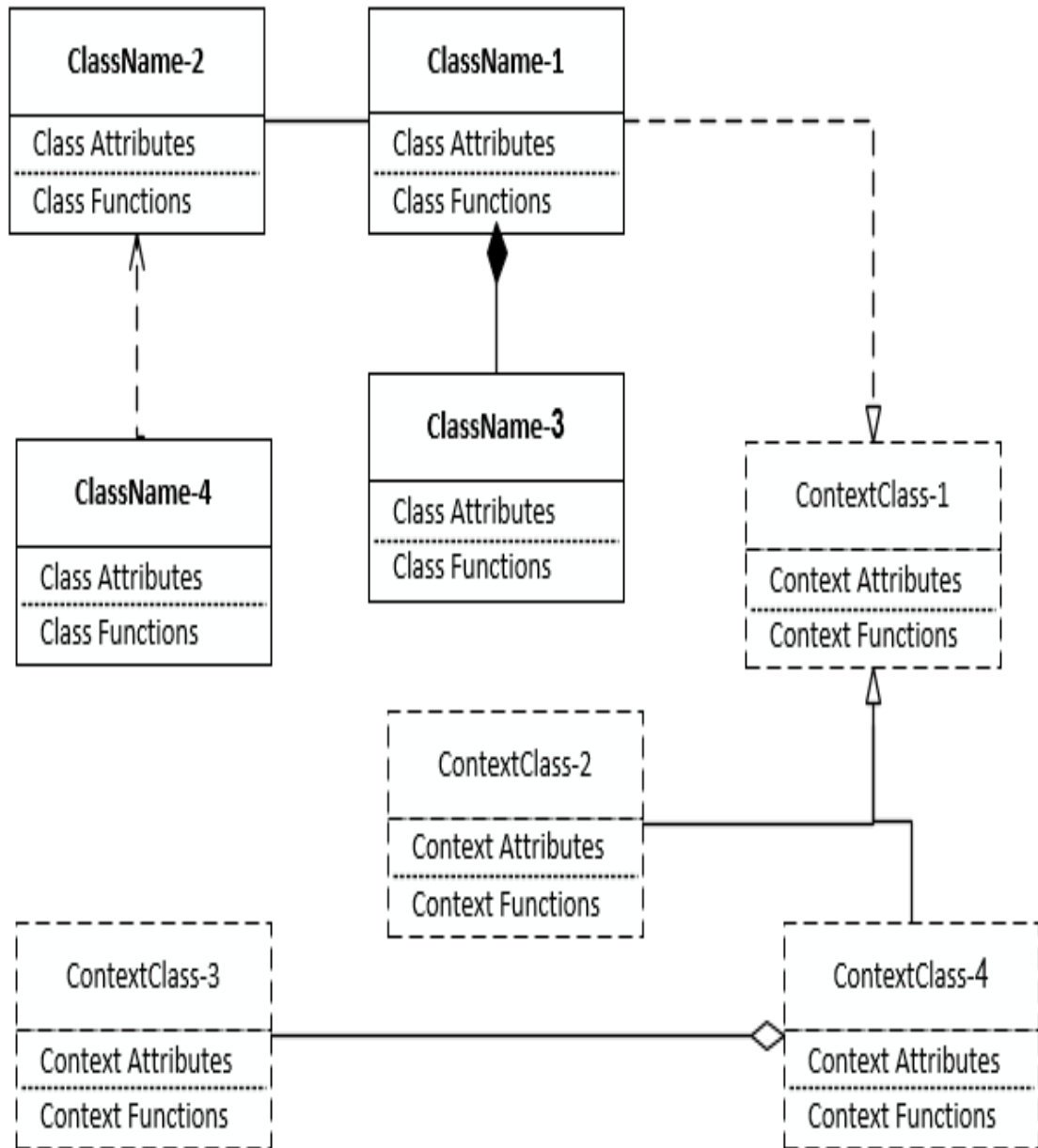


Figure 5.7: A standard modelling of a context aware class diagram

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

5.4.4 A practical example of a temperature control system using context-aware class diagram

The main components of temperature control system infrastructure are similar for different types of context aware systems such as CI providers, servers, mobility devices and so on. There are three main components that should be constructed for any intelligent system that needs to display context-awareness requirements, such as a temperature control system. However, the emergence of different environments and behaviours for context aware systems calls for many CSs to extract different CIs.

This section presents a practical example of a temperature control system. Traditional class diagrams are still facing challenging issues in terms of designing context aware systems. This example also specifies the context awareness requirements as the following temperature categories: cooling, air-fan, warming and heating. Moreover, the temperature control system manages different types of room temperature depending on user preferences to make real actions for users. This example investigates room temperature control and designs the main objects and their contexts using the suggested notions of the context aware class diagram, as shown in Figure 5.8.

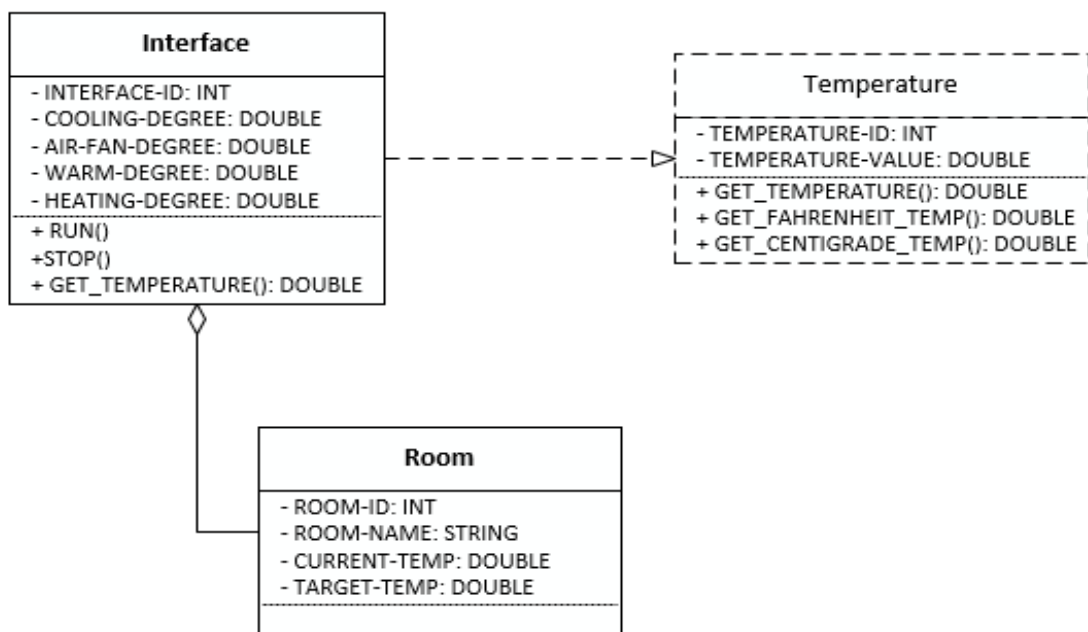


Figure 5.8: Temperature control system structure

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

5.5 Summary

This chapter has extended the class diagram to design the structure of CAAs; it has also introduced the new notion of the context-aware class diagram, which includes two concepts of the adjustable class diagram and the context class diagram. Both concepts' elements are merged for the context-aware class diagram notion to design all components of the structure of CAAs. This notion extends the class diagram for a number of reasons. The main reason is to describe the context aspects, outline the properties and functions of contexts and explain the purposes of context values to complete the CAA design stage.

In addition, this chapter 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. Moreover, a standard modelling has been designed for the static structure of CAAs, which identifies a new shape for contexts and adjusts the existing class shape for CAAs' classes so that they can express the structure of CAAs' components. Context-aware class diagram elements can design the structure of CAAs and their components: this is necessary to outline the importance of contexts and their values to design the context-awareness requirements. The values of contexts are always changing and it is important to be aware of the properties and functions of new elements. Models for CAAs have also been created to identify the specific methods for contexts which produce the values needed to fulfil the classes' needs, which affect the CAAs' services.

More importantly for CAAs' structure modelling, retrieving a set of CI values will enhance CAAs' decisions to provide different options for the user. In other words, the context-aware class diagram notion depicts the context values needed to complete CAAs' classes to implement specific operations as self-adaptive services in reaction to changes in the user's environment.

CHAPTER 5. AN EXTENSION OF CLASS DIAGRAM TO MODEL THE STRUCTURE OF CONTEXT AWARE APPLICATIONS

Furthermore, this chapter has investigated an example of a temperature control system to support the merging of the two concepts for the context-aware class diagram notion, designing a practical usage of the adjustable class diagram and the context class diagram that includes the existing elements and new elements for the static structure of CAA. The following chapter demonstrates a context aware navigation system using CA-UML.

6. Real-life Case Study of a Context-Aware Navigation System using CA-UML diagrams

Objectives:

- Evaluate the CA-UML approach in this chapter using a real case study;
- Use new concepts and their notations of CA-UML to model a context-aware navigation system;
- Define the user's needs for a context-aware navigation system;
- Specify the functional requirements and context-awareness requirements of a navigation system using the notion of the context-aware use case diagram;
- Model the dynamic behaviour of the navigation system using the notion of the context-aware activity;
- Design the static structure of the navigation system using the notion of the context-aware class diagram.

6.1 Introduction

Navigation system provides a set of services for drivers or other systems which are context-aware, connecting navigation system providers (such as GPS) with the location coordinates necessary for many systems and their users' locations, which are important CIs extracted by GPS to resolve existing systems' problems. The field of traffic modelling has become increasingly interesting, as it can help to resolve common transportation challenges around the world, but there are still numerous challenges which need further research and development in many aspects of traffic services, such as modelling, simulation, designing, auditing and implementation. The objective of traffic services is to trace the movements of traffic on specific roads, and also to compare different routes with each other to establish which road has less traffic and guide drivers and users to a faster route or direction.

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

Guidance activities by traffic services are under high demand and should be considered carefully to minimize confusion on the roads [42, 43]. This chapter investigates a case study of navigation system that includes two subsystems, the first of which constructs context awareness requirements for navigation services, while the second presents context awareness requirements for traffic services.

In addition, the main target of the CA-UML approach is to visualize the context awareness requirements to increase the navigation subsystems' productivity and the way they automatically implement the needed services to save users' time and effort and suggest options depending on users' preferences and behaviour.

In particular, the results of CA-UML present new extension modelling of the context-aware use case diagram, the context-aware activity diagram and the context-aware class diagram, merging their new notations and traditional UML notations together with the aim of increasing the quality of models, which will help to decrease the complexity of navigation system. Finally, the pragmatics and flexibility of the proposed extensions of UML diagrams are demonstrated using a real-world case study of navigation system using CA-UML diagrams.

6.2 The infrastructure of a Context-Aware Navigation System

Navigation systems have become popular around the world, as they can help users to find any location, but they also provide other services. For a better understanding, this thesis considers navigation system as an example of CAA to express the CA-UML approach. The main idea behind navigation system is awareness of a set of types of CIs, such as location, traffic, speed and remaining journey time. GPS is used to capture these CIs, as it is the most advanced communication network in the world, supporting aircraft, cars and so on [90, 113, 116]. Users can achieve navigation system services through different devices such as smart phones, laptops, PDAs and special navigation devices. The three levels of infrastructure for navigation system are shown in Figure 6.1.

**CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS**

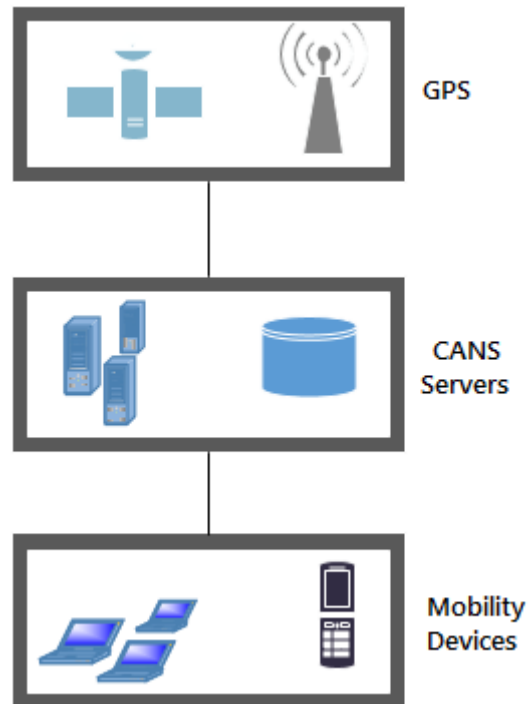


Figure 6.1: Context-aware navigation system Infrastructure

The navigation system is principally comprised of three aspects as follows:

- CI Provider

This element is used to collect information on specific entities and transfer it from physical to virtual form. A smart system such as navigation system collects the parameters by itself, using different CSs such as GPS to produce specific services and manage them as self-adaptive services.

- Navigation system Server

This element is used to manage system services and operations, and also storage and databases, to save all system data and transactions to support users depending on their requests and needs. Retrieval of information and its changes calls for different CSs to monitor specific CIs needed for users to produce the correct services.

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

- Navigation system Mobility

There are a set of types of device with the ability to use available navigation system services such as laptops, smart phones and other devices to display functional requirements and context awareness requirements depending on user preferences, which indicate what information is stored in the user profile.

6.3 Modelling the Navigation Subsystem using CA-UML diagrams

Navigation Systems currently run a series of global services to produce a virtual platform guide for users. The main CS is GPS, which is used to provide different types of CIs for navigation system that are used for various purposes such as identifying the user's location and road monitoring, and which also provide a variety of applications and update changes to CIs. The behaviour and structure of navigation system using CA-UML diagrams includes the context-aware use case diagram, context-aware activity diagram and context-aware class diagram notions examined in this thesis.

In addition, GPS is used for sensing and monitoring important objects such as location, direction, speed and so on. Reading and sending of messages through the GPS network is necessary to monitor the movement from node to node to follow and update these changes. Furthermore, several CSs aim to evaluate the monitoring procedure and follow up the CIs' changes, as illustrated in this chapter. The user can access information from the Navigation System Databases relating to locations, although such a service would require interaction with different Context Sources. There are many traditional systems that are unable to interact or communicate with context sources and sensors; also, interaction may not occur with main system functionalities but occasionally with context services [12].

Navigation Subsystems are useful for many users, especially drivers and pilots, to find different destinations, and their use is becoming increasingly common. The main tasks

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

of a navigation subsystem are to provide directions; suggest alternative directions in circumstances such as losing one's way, an accident or a blocked road; to provide the approximate remaining time to the user's destination and show the current position of the user and the nearby points of interest.

In addition, navigation system estimates the remaining time for each path period depending on the road situation and the vehicle's speed. The remaining time cannot be stable from the start of the journey until the destination for many reasons, such as heavy traffic, breakdowns, driving time regulations, rest periods and speed limits.

6.3.1 Specifying the functionalities of the Navigation Subsystem using the context-aware use case diagram notion

This section provides the main functional requirements of navigation subsystems and uses the concept of the adjustable use case diagram through the traditional notations of the use case diagram but for different usages to fulfil the functional requirements of the navigation subsystem. It also provides the main context awareness requirements of navigation subsystems and uses the concept of the use case diagram through new notations of the use context diagram to complete the functional requirements by values of CIs; also, the functional requirements and context awareness requirements of the navigation subsystem will be explained in detail using a context-aware usage scenario template. In addition, this section will explain the functional requirements and context awareness requirements using the concepts of the use case diagram and the use context diagram and those requirements will be depicted together using the notion of the context-aware use case diagram.

- Adjustable Use Case Diagram

The functional requirements of the navigation subsystem are depicted in this research using an adjustable use case diagram using the same notations and their relationships with the use case diagram. These functional requirements are user preferences, 'know

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

nearby oil stations’, ‘get directions’ and ‘know remaining time’. Those requirements are explained and depicted as follows:

- <User preferences> are the most important functional requirements for any CAA, which control the user profile in CAAs to outline the available options for searching, notifications, selections and others. Large numbers of users request flexibility to set more preferences to get specific services: user preferences are important, as they allow users to insert all of their requirements and needs. The interface of the navigation subsystem should increase the options for users to achieve more control over navigation services (as depicted in Figure 6.2.A). User preferences are functional requirements in navigation system to process the functions. For example, the postcode of the destination is one of the user preferences and the user should insert it into the navigation system to achieve the final destination. <Know nearby oil station> is an example of a user preference that allows users to search for nearby oil stations.

The relationships between <user preferences> and <know nearby oil station> is <<extend>>, meaning that this is an optional service and the driver may or may not use it.



Figure 6.2.A: Functional requirement of user preferences for navigation subsystem

- <Know faster path> is one of the services of the navigation subsystem; this selection of a faster path needs CI values to calculate the value of the remaining time and then select the smallest value (quickest way to get to the destination). Figure 6.2.B shows an example of the navigation subsystem of <Know faster path> (as shown in Figure 6.2.B); this is not a normal system and requires more notations to illustrate context awareness requirements and

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS

other activities to make any task. This does not mean that CAAs cannot use the existing use case diagram notations, but they still need an extension concept of the use case diagram and more notations to explain the context awareness requirements to create clear and accurate models.

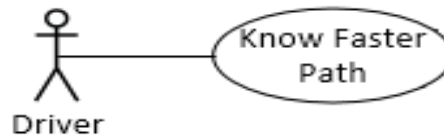


Figure 6.2.B: Functional requirement of know faster path for navigation subsystem

- <Know remaining time> and <Know speed> are important functional requirements of the navigation subsystem (as depicted in Figure 6.2.C) but need to interact with the traffic subsystem to inform the driver of the real duration by calculating the remaining journey time and give notifications of upcoming breaks affecting the remaining journey time.

However, this service can be affected by the amount of traffic on the road, and the remaining time is constantly changing depending on the road situation.

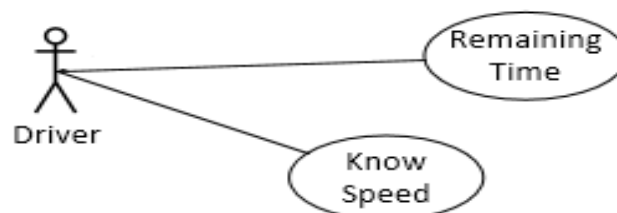


Figure 6.2.C: Functional requirements of know remaining time and speed for navigation subsystem

- <Get position> is a common functional requirement in navigation subsystems and is used to specify X, Y coordinates of the current position or final position (destination) and give multiple options about journey directions. This helps the driver to select the most suitable directions to

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

achieve the final destination and also provides more details about the driver's current position and journey direction, such as how many miles in each direction. However, render maps specify a suitable map for the driver depending on the coordinates of the current position or final position. In addition, <Get position> displays the position of the driver on the map and optionally may be extended to include the points of interest in the vicinity (as shown in Figure 6.2.D).

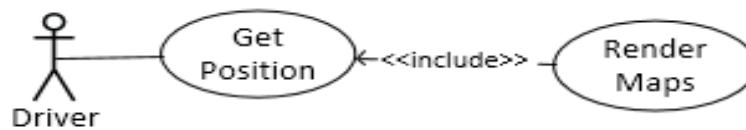


Figure 6.2.D: Functional requirement of Get position for navigation subsystem

- Use Context Diagram

The context awareness requirements of the navigation subsystem are depicted in this research by the use context diagram using new notations and their relationships, including different types of CIs.

The context awareness requirements of the navigation subsystem are speed, location, coordinates, time to destination, nearby location, distance, speed limits, direction and map data as follows:

- <Distance> and <Speed> are composite CIs to calculate the value of remaining time and important context awareness requirements to carry the values of CIs in the navigation subsystem used. Those context awareness requirements are depicted as use context notations to specify accurate output and complete functional requirements of the known remaining time or known elapsed time of such a journey. Those CIs are extracted by different CSs such as GPS and WhereAmI Service (as shown in Figure 6.3.A).

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS



Figure 6.3.A: Context awareness requirements of distance and speed

- <Map data> and <Coordinates> are composite CIs to calculate the value of the render map and mandatory values for context awareness requirements to complete the functional requirement of <get position> for the driver in the navigation subsystem. Those context awareness requirements are depicted as use contexts notations. The render map draws maps for the user to get to a specific position (as shown in Figure 6.3.B).



Figure 6.3.B: Context awareness requirements of map data and coordinates

- <Speed limits>, <Coordinates> and <Time to destination> are composite CIs to calculate faster routes for drivers. Those context awareness requirements are depicted as use context notations (as shown in Figure 6.3.C). The CIs are needed to fulfil the functional requirement of 'know faster path' and are used to draw options to specify which path between one place and another is retrieved from the navigation system (navigation system includes a render engine to produce maps).

**CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS**

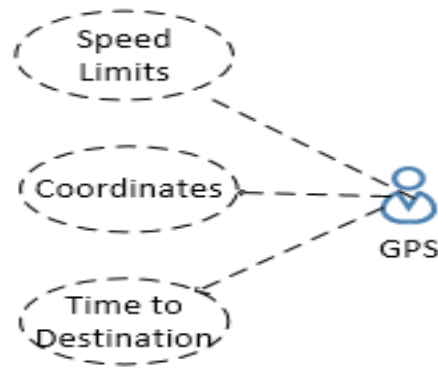


Figure 6.3.C: Context awareness requirements of speed limits, time to destination and coordinates for navigation subsystem

- Context Aware Use-case Diagram

Navigation subsystems are important when it is necessary to take a journey in the least possible time using the most accurate and current method to reach the destination. In order for the navigation system to be accurate, the following information is needed: start point of journey, end point of journey, direction of journey using the most convenient route, and distance of journey with remaining time.

The context-aware use case diagram notion shows the use cases corresponding to these functionalities, the use contexts representing the CIs upon which these functionalities depend and the CSs that will provide the raw data for computing these CIs.

It is assumed that the map service (such as Google Maps) provides all the data necessary to render the road network, including location coordinates of types of road.

In order to render the driver position and possibly the nearby locations needed for drivers, 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.

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS

Table 6.1: Summary of functional requirement and context awareness requirement for navigation subsystem

Navigation Subsystem		
Functional Requirements	Context-awareness Requirements	CI Provider
Know location	- distance	- WhereAmI Service
Remaining time	- distance - speed	- GPS - WhereAmI Service
Get position	- coordinates - map data	- GPS - Map Service
Know speed camera	- speed limits	- GPS
Know nearby oil station	- nearby location	- GPS
Know faster path	- speed limits - coordinates - time to destination	- GPS

Table 6.1 shows a summary of the functionalities of the navigation subsystem, while Figure 6.4 depicts the main requirements in this case study using the context-aware use case diagram notion, including both concepts and their notations for the adjustable use

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

case diagram and the use context diagram, as well as identifying the practical usage of each CI required for navigation system and which CS providers are available.

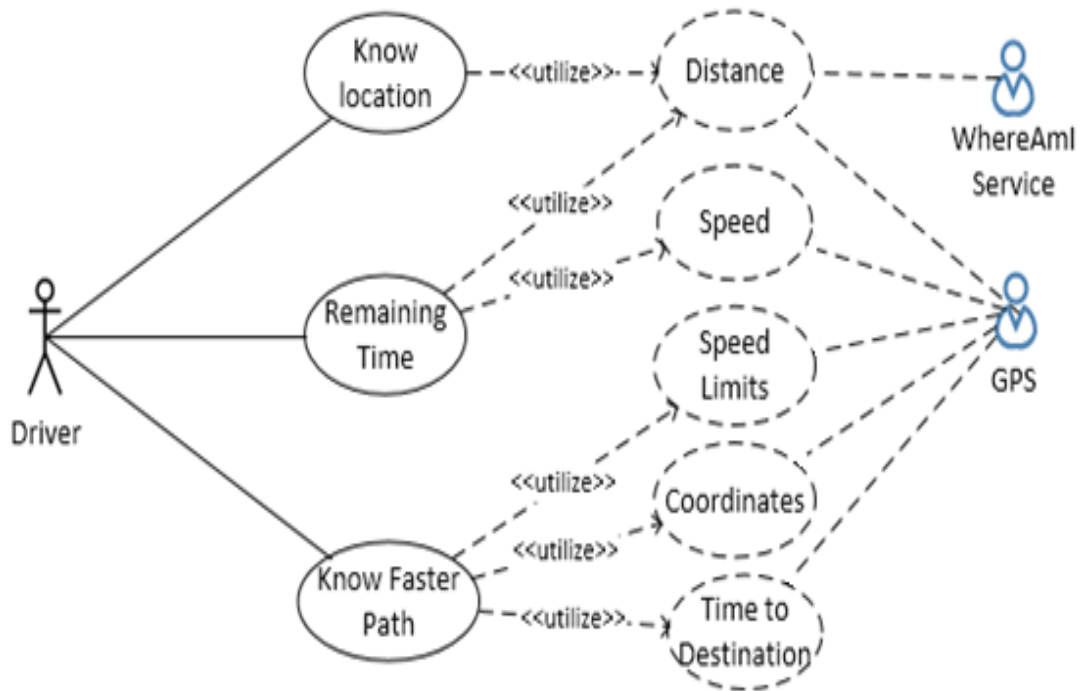


Figure 6.4: A navigation subsystem

In addition, the navigation subsystem shows details of arrival at the destination, based on the remaining time to reach the destination: this will help users to select the appropriate route to their destination, notify them of their current location, update them on location changes and redirect them should they get lost.

- Context Aware Usage Scenario Template

The context-aware usage scenario template is an extension concept of the usage scenario template that is fully described in various stories and scenarios. Its functional requirements and context awareness requirements can be described in sequence scenarios and stories.

The context-aware usage scenario template concept describes the context-aware use case diagrams in clear English sentences: these templates help navigation system developers to understand all navigation system requirements and drivers' needs. Table

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

6.2 documents the requirements of user preferences which explain all scenarios of functional requirements and context awareness requirements for user preferences of a navigation subsystem.

Table 6.2: Functional requirement and context awareness requirement of navigation subsystem using context-aware usage scenario template

Elements	Example
Use Case Name	- User preferences - Know nearby oil station
Use Context Name	- Nearby location
Use Case ID	- User preferences: 11 - Know nearby oil station: 12
Use Context ID	- Nearby location: A11
Context-aware Use Case Diagram	<pre> graph TD Driver[Driver] --- UP((User Preferences)) UP -.-> <<extend>> KNOS((Know Nearby Oil Station)) KNOS -.-> <<utilize>> NL((Nearby Location)) NL -.-> GPS[GPS] style NL stroke-dasharray: 5 5 </pre>
Use Case Priority	Medium

**CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS**

Use Context Priority	Medium
Use Case-11 Scenario narrative	<ul style="list-style-type: none"> - The driver uses the navigation system interface - Navigation subsystem will display the main screen - The driver inserts mandatory preferences such as post-code of destination - If the driver inserts correct post-code - Navigation subsystem should retrieve the destination details from navigation system databases - The driver can select an available service - Else the system exits the interface screen
Use Case-12 Scenario narrative	<ul style="list-style-type: none"> - The driver can find out about nearby oil stations - The driver has an optional service to get to the destination of the nearest oil station - Else the driver can disable the service to know nearby oil station
Use Context-A11 Scenario narrative	<ul style="list-style-type: none"> - The driver needs to be aware of nearby locations when driving - Navigation subsystem has the ability to capture the nearest locations using different types of CSs - Navigation subsystem notifies the driver of nearby locations such as oil stations or speed cameras - Navigation subsystem suggests other locations such as hotels, restaurants and so on
Actor	Driver

**CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS**

Use Case-11 conditions	Driver must insert the post-code of the destination
Use Case-12 conditions	None (optional service to know nearby oil station)
Use Context-A11 Constraints	<ul style="list-style-type: none"> - If remaining distance of nearby location (oil station) is more than one mile - No action - Else notify driver of nearby oil station
Frequency	Multiple uses per visit
Awareness Requirements	Update driver about future events during journey time, such as road closure and accidents
Context Source Name	GPS
Alternative Context Source Name	WhereAmI Service

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

6.3.2 Modelling the dynamic behaviour of the Navigation Subsystem using the context-aware activity diagram notion

Navigation system includes two subsystems for navigation and traffic. navigation system provides important information for users (such as drivers in this case study): navigation system services specify the user's start point, destination point and journey path using the most convenient route and the remaining time for the journey.

The behaviour of navigation system needs to be modelled in simulation maps to outline the important services and reaction behaviour in different situations to guide and direct users without errors or delays. The use of smart devices by many users around the world and the fact that transportation has become faster and more comfortable for drivers means that navigation system services make it easier than ever to reach one's required destination. The most popular provider of CIs is GPS, which works for navigation system and other systems. GPS is a huge CS used to produce required CIs for navigation system and also to monitor changes, which carry many types of CIs into different applications and systems. In addition, GPS is a popular CS, extracting necessary CIs worldwide. The main CI extracted 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.

Moreover, the navigation subsystem has a database of maps that can be used to search for information about an address or location; the device system services also function to direct the user to his or her required destination. However, we need to identify the difference between navigation subsystem services and information. The user does not always use the application services and may only occasionally require information without application services such as full details about the home address of the destination (this kind of information is retrieved from system databases).

- Adjustable Activity Diagram

The concept of adjustable activity diagrams and their notations can be adjusted for this case study using basic activity diagram notations to express the behaviour of the

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

navigation subsystem. In addition, the activity diagram notations can model the behaviour of the navigation subsystem but not the adaptation actions and their constraints, which need new notations. Using the same notations that were previously used in the traditional activity diagram, it is possible to describe the work-flow between navigation subsystem activities. Also, traditional notations can model the activities of CSs such as GPS activities and their processes (as shown in Figure 6.5.A).

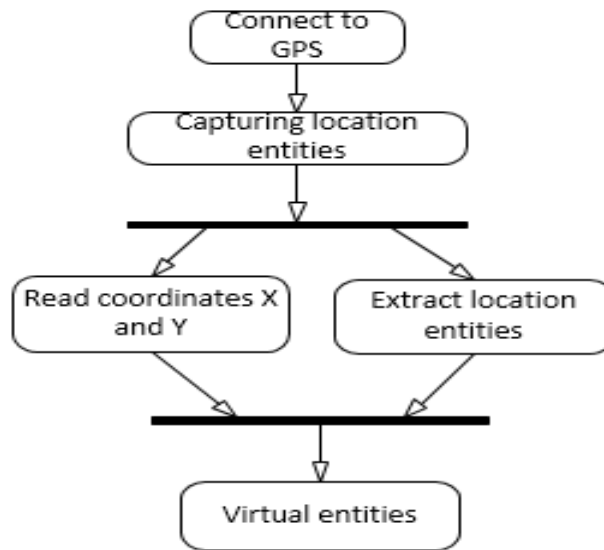


Figure 6.5.A: An example to show the notations of the adjustable activity diagram

In Figure 6.5.A the traditional notations of the activity diagram can be used to model the CAA behaviour and context acquisition to determine the activities of CSs such as GPS, to specify suitable software for the GPS and ensure that data output is provided by the GPS to identify the task queries and how they can manage and save changed CIs such as location and their coordinates.

- Context Activity Diagram

The new concept of the context activity diagram was introduced in Chapter 4, which includes new notations that help to present the dynamic behaviour of navigation subsystems, specifying the change data captured by GPS that affects adaptation actions and context constraints, helping to control the end outputs for drivers to respond to their

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

needs. This concept simplified the complexity of navigation subsystem to provide the effectiveness of activities and their constraints in clear models and diagrams.

This will fulfil the drivers' needs and outline the activities affected to produce precise services depending on user preferences and situations (as shown in Figure 6.5.B). The diagram below supports the context activity diagram concept, which uses new notations to specify the faster path: we need an if-statement function (context constraint) to calculate the smallest value to follow the faster path needed for the driver.

Node 1 is the driver's location and Node 2 is the destination. The path is to specify the distance from Node 1 to Node 2. In this case study, the driver can use three paths to reach the destination and has selected the faster path. The navigation subsystem will compare the three paths and select the smallest value as follows:

- If ($\text{Path-1} < \text{Path-2}$ and $\text{Path-1} < \text{Path-3}$), the smallest value is Path 1.
- Else If ($\text{Path-2} < \text{Path-1}$ and $\text{Path-2} < \text{Path-3}$), the smallest value is Path 2.
- Else the smallest value is Path 3.

To calculate distance values, the navigation subsystem needs the driver coordinate values for X and Y, which are extracted by GPS and carried to the server of the navigation subsystem.

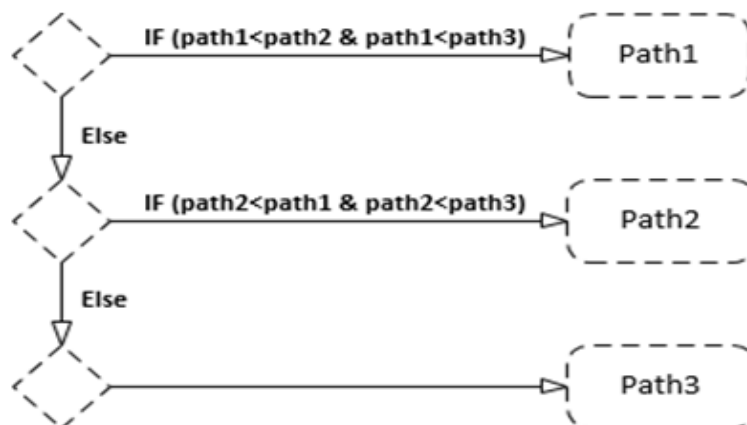


Figure 6.5.B: An example to show the notations of the context activity diagram

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

- Context Aware Activity Diagram

In this case study, the context-aware activity diagram notion considers three categories: the driver, the navigation subsystem, and the GPS. The activities and the interactions between them will lead to the overall behaviour modelling. However, the context-aware activity diagram notion provides two levels to model CAAs; this case study will use both of them to support the context-aware activity diagram notion using new notations and swimlanes for separation between categories. Also meta-swimlanes are used to partition each category into different parts and express their interactions. In this case study, a standard modelling of high-level interactions will be used to model the dynamic behaviour of the traffic subsystem in the following section, while a standard modelling of low-level interactions will be used for the navigation subsystem in Figure 6.5.C, which depicts the behaviour of the navigation subsystem and the interaction with the traffic subsystem to express overall activities of navigation system.

Figure 6.5.C classifies navigation system into three main swimlanes for the driver, navigation and traffic subsystems and the GPS and traffic service. The driver swimlane includes meta-swimlanes for two drivers' activities which require a secure login for Adam only. Security activities within the driver meta-swimlane include three functionalities required for navigation system to verify the driver authentication: the first activity (Login) is to verify that the driver's name and password are correct, while the second activity is to check the account's validity to access navigation system services, such as whether the account needs to be renewed to allow longer or unlimited access depending on the user's contract with the provider. The third activity is to check whether the user would like navigation system to remember his account login details to allow him to access navigation system again later without having to re-enter his user name and password, although the user can decline this option for greater privacy. The driver's position and destination are the most important CIs used in the GPS to specify the user preferences for the location details that are required to inform the navigation subsystem. The most important data is usually the postcode, which is required by databases as the primary key for retrieval purposes to find the full details, such as position details, coordinates and the direction of the destination.

**CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS**

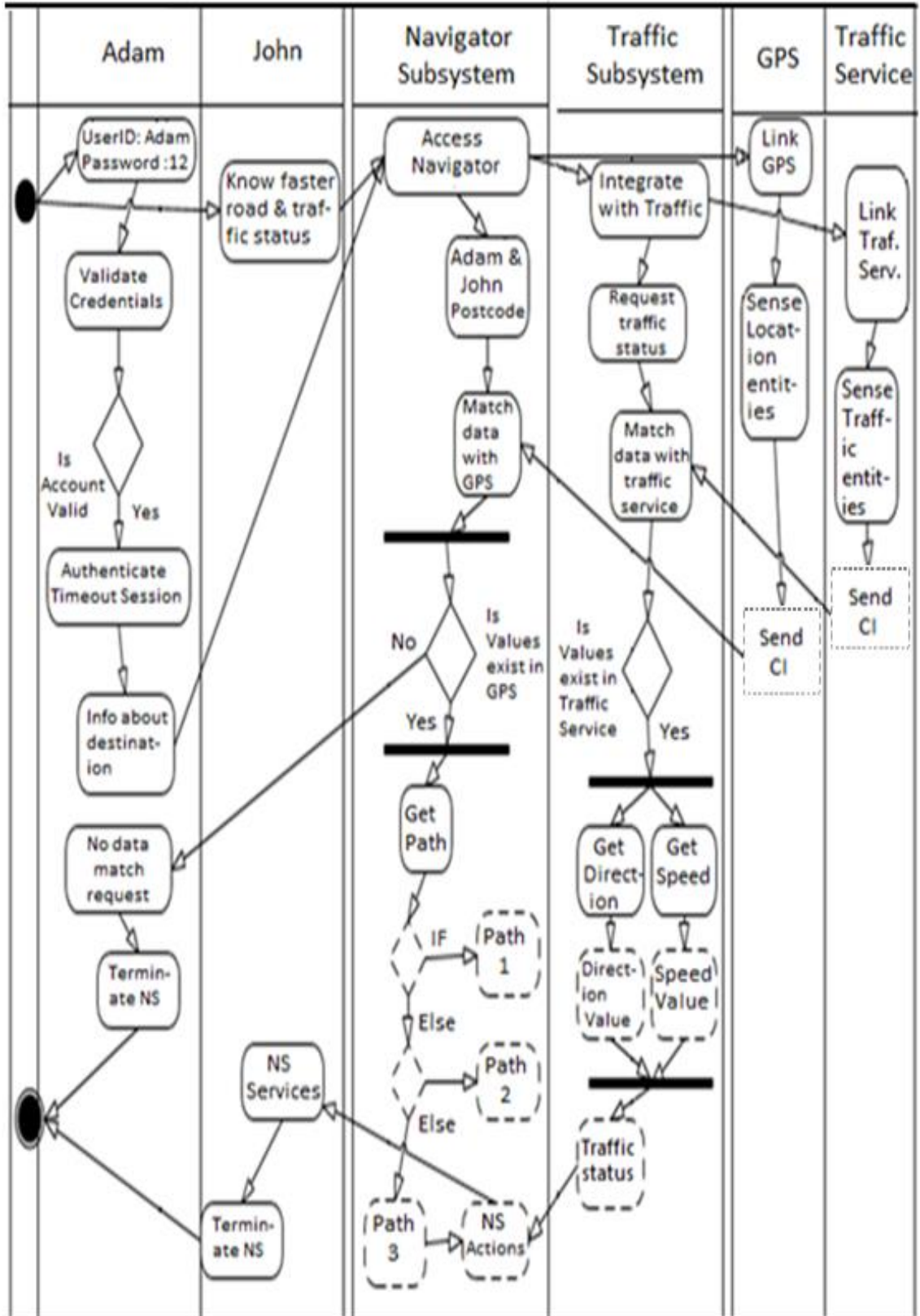


Figure 6.5.C: The main activities of the navigation and traffic subsystems

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

Also John specified user preferences (as shown in Figure 6.5.C) to achieve a faster path and find out traffic information, and can access the other services of navigation and traffic subsystems as well as the activities involved in identifying the fastest route: this is a good example to show the new notations of context constraints and adaptation actions which determine the navigation system conditions to make the required adaptation action. Adam inserted an unfound postcode into the navigation system services: this will produce unavailable services for Adam and he will need to check the postcode with the provider of the navigation system.

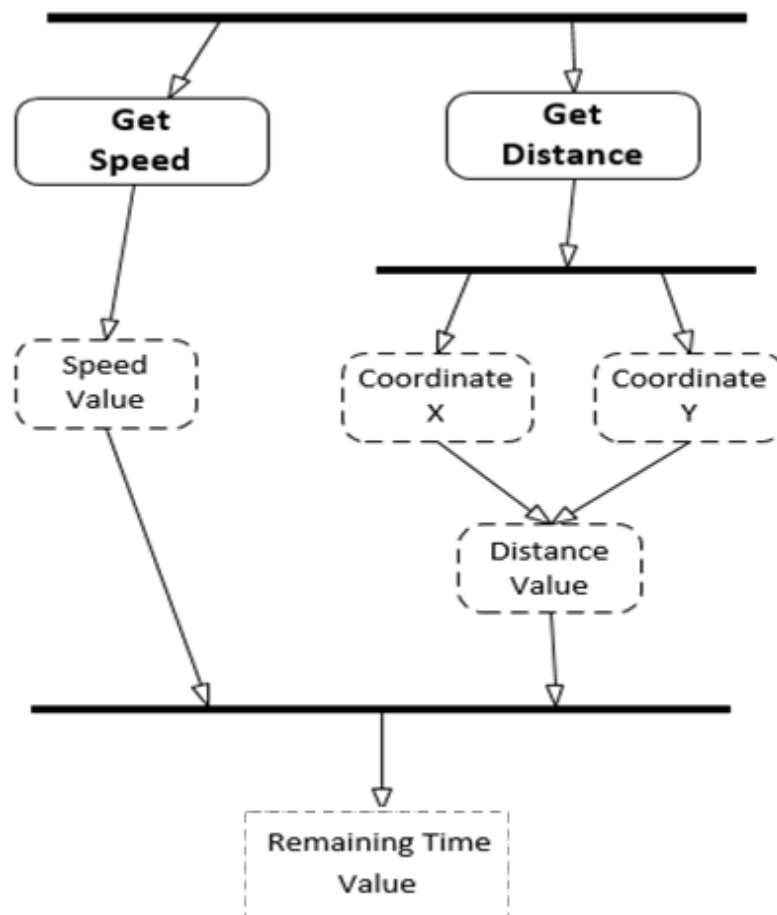


Figure 6.5.D: An illustration of remaining time activities

Figure 6.5.D depicts the remaining time using the context-aware activity diagram notion to provide a diagram expressing the activities of remaining time, which needs a function

**CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS**

to calculate the distance value and the speed value. All drivers' needs should be specified in the user preferences. The context-aware activity diagram notion provides behaviour modelling to depict the interaction of navigation system activities, context acquisition processes and adaption actions to notify the final clients of navigation system services.

Table 6.3: Summary of behaviour activities for navigation subsystem

Navigation subsystem	John
Adaptation Action	Faster path and traffic information
Acquisition Sense	GPS and Traffic service produce the driver's location, distance and speed
CI Type	Atomic CI: path, distance and speed Composite CI: remaining time and traffic information
Constraint	- Path 1 if smallest - Else path 2 - Else path 3

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

6.3.3 Designing the static structure of the Navigation Subsystem using the context-aware class diagram notion

The services of the navigation subsystem are a series of global services as a standard system worldwide to produce a virtual platform guide for drivers. The main provider of CIs is GPS, which is used to provide navigation subsystems for monitoring a variety of systems and updating changes to CIs. Additionally, GPS is used for sensing and monitoring important objects such as location, direction, speed and other objects.

The reading and sending of 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.

- Adjustable Class Diagram

Traditional elements of the class diagram have the ability to design the main context objects of the navigation subsystem (as shown in Figure 6.6.A). There are important context objects that should be designed in the design stage of the navigation subsystem, such as acquiring duration updates for drivers and rendering a map for retrieving the appropriate map for a specific journey.

These need many single CIs: speed, direction, position coordinates and map data and details, which are provided by GPS or web services such as the *WhereAmI* service.

**CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS**

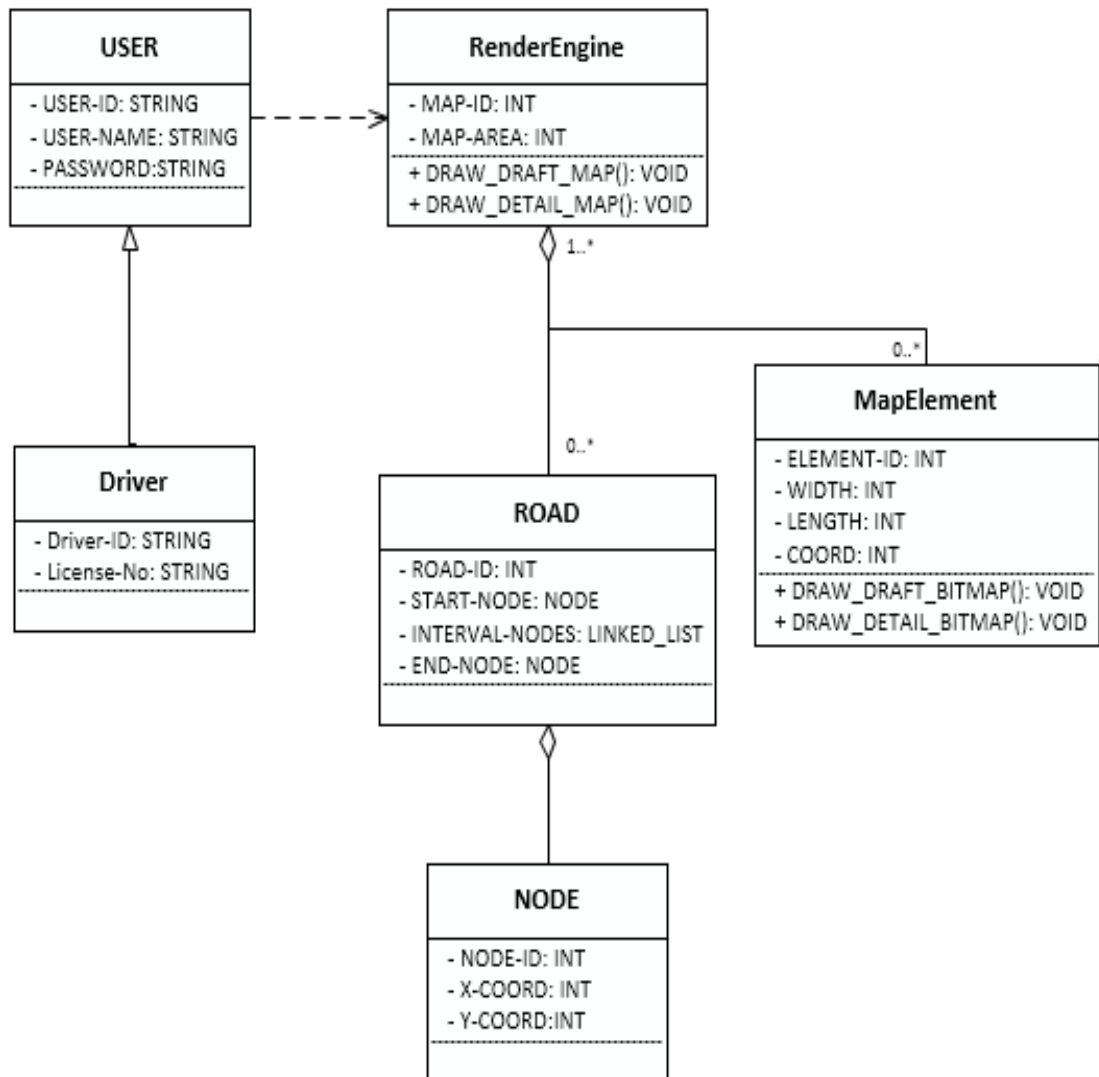


Figure 6.6.A: An example to show the elements of the adjustable class diagram

- Context Class Diagram

New elements of the context class diagram are introduced in Chapter 5, and are used to design the structure components of contexts and their entities to complete the functions of the navigation subsystem (as illustrated in Figure 6.6.B). The navigation subsystem is an example of a CAA which displays smart services and informs drivers as a self-adapting system using CSs such as GPS, which is a popular context service, but not a CAA used to specify driver location and other CIs.

**CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS**

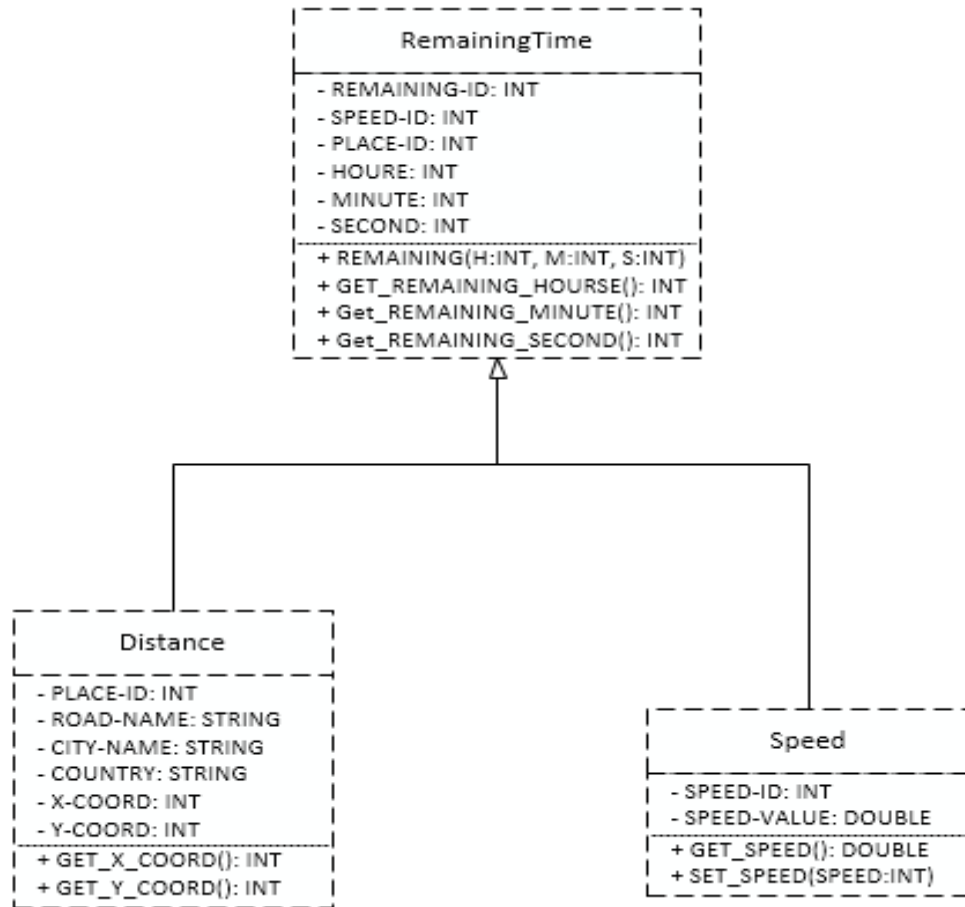


Figure 6.6.B: An example to show the elements of the context class diagram

- Context-Aware Class Diagram

The classes and contexts in Figure 6.6.C are designed using the context-aware class diagram notion (elements of the adjustable class diagram and the context class diagram are merged) to show the main components of the navigation subsystem. Figure 6.6.C shows how the context-aware class diagram notion demonstrates the main facilities of a navigation subsystem which updates its parameters using different providers of CIs and their changes depending on the driver's environment. In addition, a new relationship called 'utilization' is used to retrieve the values of contexts to fulfil the object requirements of the navigation subsystem. The following diagrams illustrate the context-aware class diagram notion for a navigation subsystem and its components.

**CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS**

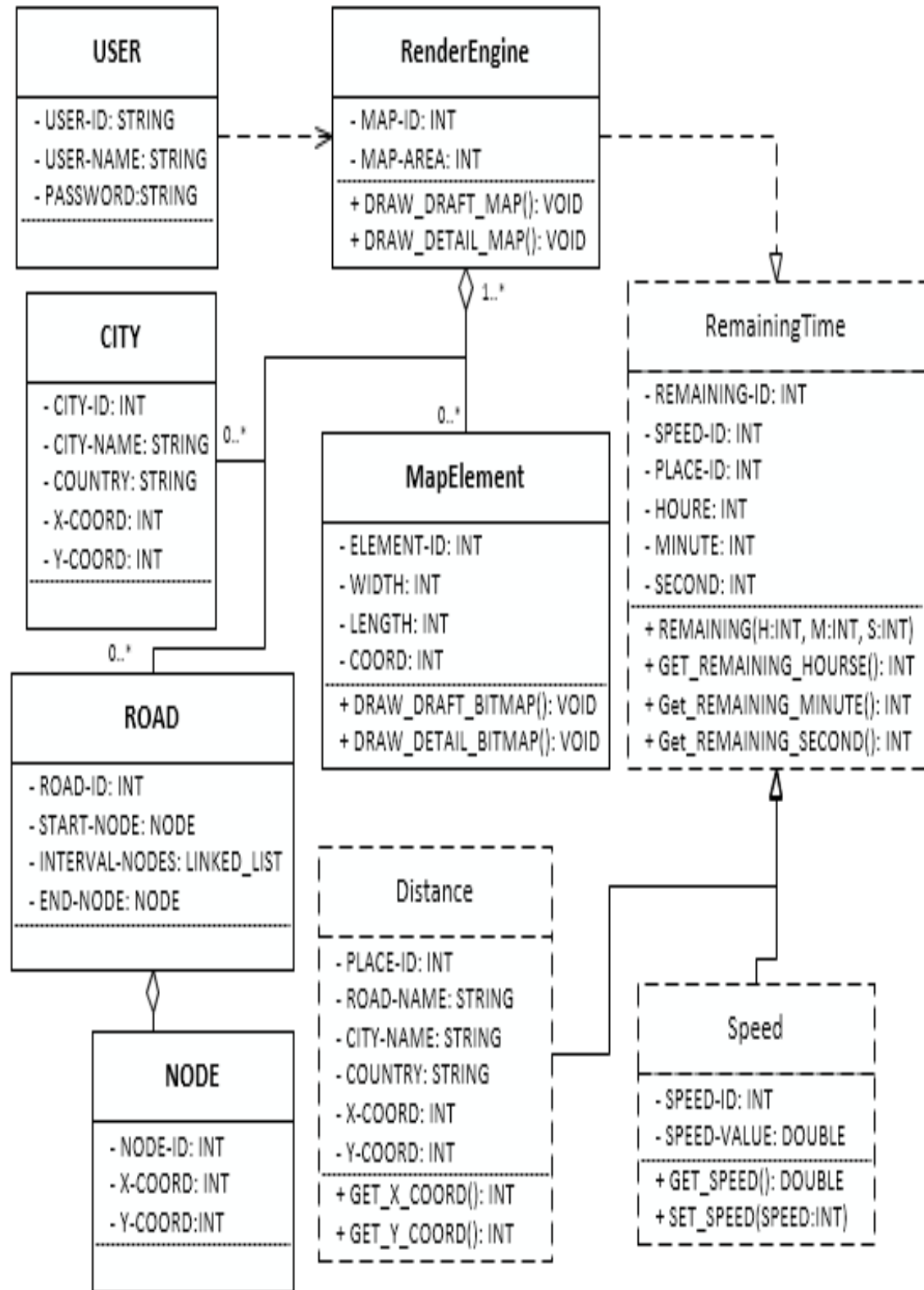


Figure 6.6.C: The static structure of the navigation subsystem

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

The above diagram (Figure 6.6.C) illustrates the structure components of objects and contexts for the navigation subsystem using the elements of the context-aware class diagram notion and their relationships. A number of different relationships are used in this notion, such as the generalization relationship for inheritance purposes.

A new relationship of utilization is used to fulfil the context awareness requirements of the navigation subsystem and complete the design stage to outline which CIs are needed and their properties and operations.

6.4 Modelling the Traffic Subsystem using CA-UML diagrams

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 subsystem to alert drivers to road situations.

The traffic subsystem provides traffic information such as speed, speed limits, and other services about road situations and movements. This situation is important, enabling it to alert users to road conditions using different types of sensors (usually traffic affects navigation system services which affect the remaining journey time).

6.4.1 Specifying the functionalities of the Traffic Subsystem using the context-aware use case diagram notion

Traffic functionalities provide a high level of reliability and performance to clarify the traffic flows along the route that might impact the driver and ensure all network junctions for drivers of different vehicles such as private cars, taxis, bikes, public vehicles and bus drivers.

This section provides the main functional requirements of the traffic subsystem using the concept of the adjustable use case diagram through traditional notations of the use case diagram but for a different purpose. It also provides the main context-awareness requirements of the traffic subsystem and uses the concept of the use context diagram and its notations to complete the functional requirements using values of CIs. The functional requirements and context awareness requirements of the traffic subsystem will be explained in detail using the context-aware usage scenario template concept. In addition, this section will explain the functional requirements and context awareness requirements using the concepts of both the use case diagram and the use context

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

diagram and depict those requirements together using the notion of the context-aware use case diagram.

- Adjustable Use Case Diagram

The functional requirements of the traffic subsystem are user preferences, get traffic information, and get location (such as city or country). Those requirements are explained and depicted as follows:

- User preferences can be expanded to include any need for users. The traffic subsystem provides more functional requirements, which increase navigation system services, such as obtaining traffic information and the speed limits on a particular road. However, drivers can specify different preferences, such as receiving notifications if there are speed cameras on the road. An example of user preferences for the traffic subsystem is <know speed camera> to help the driver to slow down when a speed camera is found on a particular road. The relationships between <user preferences> and <know speed camera> is <<extend>>, which means that it is an optional service and the driver may or may not use it (as shown in Figure 6.7.A).

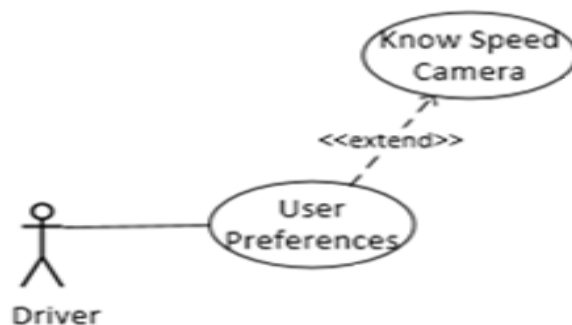


Figure 6.7.A: Functional requirement of user preferences for the traffic subsystem

- <Get traffic information> is the main functional requirement of the traffic subsystem, which is the most important service used to find traffic-related information such as speed, direction and road movement: this will produce

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS

the accurate duration of the fastest route. In addition, it calculates the speed and the time to destination using information from a GPS module or traffic service; and determines the speed limits using the location data from the GPS and the road information provided by a map service. <Know speed limit> is another functional requirement that might be needed for drivers in harsh weather conditions when it is difficult to see signs indicating speed limits, especially on motorways (as shown in Figure 6.7.B).

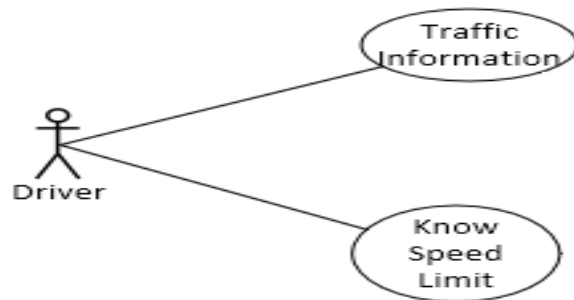


Figure 6.7.B: Functional requirements of traffic information and know speed limit

- <Suggest alternative direction> – this functional requirement is necessary when something happens suddenly and causes road movement to slow down, such as a car accident; this situation should inform the driver and the navigation system will suggest different directions to get to the destination via a quicker route (as shown in Figure 6.7.C).



Figure 6.7.C: Functional requirement of alternative direction

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

- Use Context Diagram

The context-awareness requirements of traffic subsystems are depicted in this case study using a new concept and notations of use context diagram notations and their relationships, including different types of CIs.

However, the context awareness requirements of the traffic subsystem are speed, location, road movement, coordinates, speed limits, direction and map data as follows:

- <Direction> and <Speed> are composite CIs to calculate the value of traffic information, which is mandatory information to fulfil context awareness requirements to carry the values in the traffic subsystem. Those context awareness requirements are depicted as use context notations to specify accurate context awareness requirements to complete the functional requirements of 'know traffic information' or 'know speed of car'. These CIs are extracted by different CSs, such as GPS and traffic services (as shown in Figure 6.8.A).

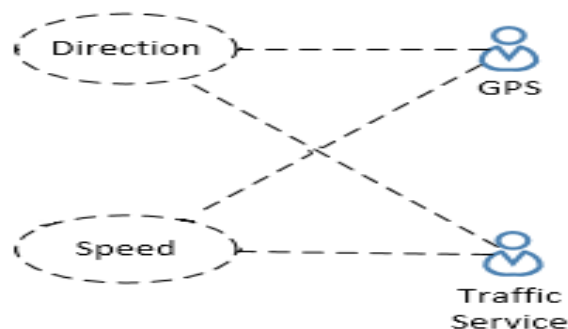


Figure 6.8.A: Context awareness requirements of direction and speed

- <Direction> and <Movement> are composite CIs and are required to make adaptation decisions for the functional requirement of <Suggest alternative direction>. The traffic subsystem also needs to be aware of the movement on certain roads. Slow movement means that the road has heavy traffic or another event has caused the road to be closed. Context awareness requirements need CI of movement and direction (as shown in Figure 6.8.B).

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS

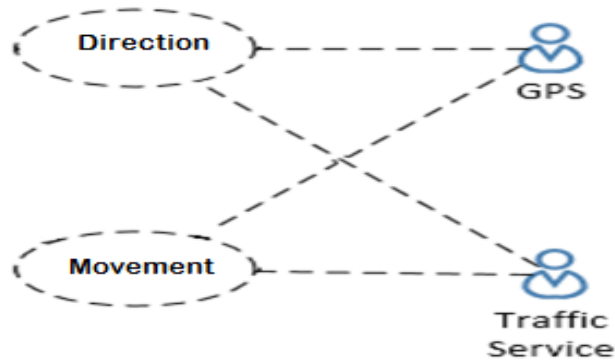


Figure 6.8.B: Context awareness requirements of direction and movement

- <Speed limits> is an atomic CI and may be needed for many uses. In the traffic subsystem, this value is required to complete the user preference to know the speed limit. However, the context awareness requirement of speed limits is necessary to complete the functional requirement of <know speed limit> (as shown in Figure 6.8.C).

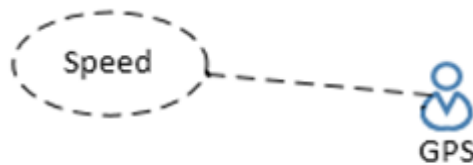


Figure 6.8.C: Context awareness requirement of speed for traffic subsystem

- Context Aware Use-case Diagram

The traffic subsystem provides many services for drivers, one of which involves getting directions, which is depicted in this case study using an adjustable use case diagram. However, the functional requirement of <get alternative directions> shows the route to the destination and optionally suggests alternative directions for drivers. Another important service is factoring in alternative options such as suggested routes in case of road traffic accidents or other unexpected events.

**CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS**

Table 6.4: Summary of functional requirements and context awareness requirements for traffic subsystem

Traffic Subsystem		
Functional requirement	Atomic CI	CI Provider
Know speed limits	- Speed limits	- GPS
Suggest alternative directions	- Movement - Direction	- GPS - Traffic Service
Get traffic information	- Speed - Direction	- Traffic Service

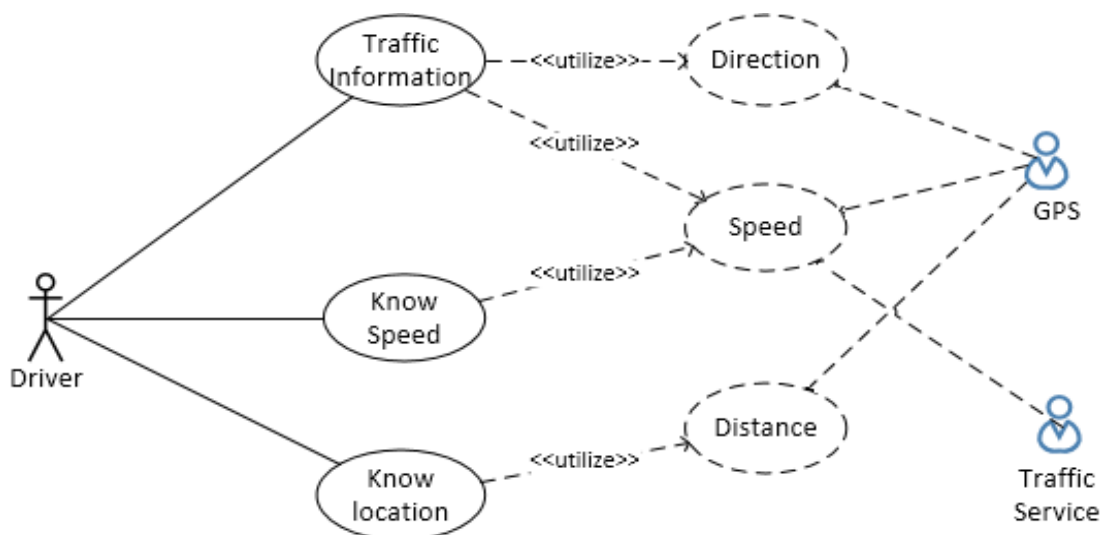


Figure 6.9: A traffic subsystem

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

- Context Aware Usage Scenario Template

The context-aware usage scenario template concept is used in this case study to explain the context-aware use case diagrams in sequence sentences, and also to document the requirements of the traffic subsystem and clarify them in such a way as to enable designers to implement all requirements and meet drivers' needs. Table 6.5 describes the requirements of user preferences and outlines all scenarios of functional requirements and context awareness requirements for user preferences within the traffic subsystem. In addition, the concept of the context-aware usage scenario template investigates the conditions and constraints of the traffic subsystem and finds solutions to special issues to guarantee navigation system quality.

Table 6.5: Functional requirement and context awareness requirement of traffic subsystem using context-aware usage scenario template

Elements	Example
Use Case Name	- User preferences - Know speed camera
Use Context Name	- Speed
Use Case ID	- User preferences: 22 - Know speed camera: 23
Use Context ID	- Speed: A23

**CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS**

Context-aware use case diagram	<pre> graph LR Driver((Driver)) --- UP((User Preferences)) UP -.-> <<extend>> KSC((Know Speed Camera)) KSC -.-> <<utilize>> Speed(((Speed))) Speed -.-> GPS((GPS)) </pre>
Use Case Priority	Medium
Use Context Priority	Medium
Use Case-22 Scenario narrative	<ul style="list-style-type: none"> - The driver uses the navigation system interface - Traffic subsystem will display on the main screen - The driver inserts mandatory preferences such as the post-code of the destination - If the driver inserts the correct post-code - Traffic subsystem should monitor the selected direction and notify drivers of road traffic or sudden events - The driver can select an available service - Else the system exits the interface screen
Use Case-23	<ul style="list-style-type: none"> - The driver can find out about speed cameras - The driver has an optional service to be alerted to the locations of

**CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS**

Scenario narrative	<p>speed cameras along certain roads</p> <ul style="list-style-type: none"> - Else the driver can disable the service to know about speed cameras
Use Context-A23 Scenario narrative	<ul style="list-style-type: none"> - The driver needs to be aware of speed limits when driving - Traffic subsystem has the ability to capture speed limits using different types of CSs - Traffic subsystem notifies the driver of speed limits - Traffic subsystem suggests to slow down when car speed exceeds the speed limit
Actor	Driver
Use Case-22 conditions	Driver must insert the postcode of the destination
Use Case-23 conditions	None (optional service to know speed camera)
Use Context-A23 Constraints	<ul style="list-style-type: none"> - If speed less than speed limit - No action - Else notify driver to slow down
Frequency	Multiple uses per visit
Awareness	Update driver of future events during journey time, such as road

**CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS**

Requirements	closure, slow movement and accidents
Context Source Name	GPS
Alternative Context Source Name	Traffic Service

6.4.2 Modelling the dynamic behaviour of Traffic Subsystems using the context-aware activity diagram notion

Traffic services are affected by environment changes (such as road maintenance or car accidents) and the modelling stage is important to alert the driver to the impacts of traffic and to suggest different plans, which will enable the driver to resolve any future impact immediately, as well as to specify special behaviour requirements and investigate future needs.

Traffic is information and is considered by roads, with the output result of traffic movement and flows. CAA requirements include traffic information as CI, which is important information for many uses (such as remaining time on journeys affected by traffic). In addition, traffic information supports other systems and applications (such as navigation subsystems). For example, remaining journey time is one functional requirement of the navigation subsystem that can be calculated without traffic information, but the result will be not accurate and the journey can be delayed if there is heavy traffic. Furthermore, traffic services provide continuous information on journey durations, outline the road situation early and suggest available journey routes.

This case study of navigation system using the context-aware activity diagram notion demonstrates the main activities of both subsystems – navigation and traffic – which

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

update themselves, depending on available CIs and their providers, and usually change depending on the driver's environment.

The traffic subsystem is an example of a CAA informing drivers using different types of CSs, such as GPS and traffic services; CS is a CI provider, not a CAA used within a location (the most common CI) based on a CAA acting as a sensor.

- Adjustable Activity Diagram

The interaction activities between the driver, the traffic subsystem and the traffic service are illustrated using the basic notations of the adjustable activity diagram.

This case study of navigation system used two subsystems to express how to depict the activities between the two subsystems and used the context-aware activity diagram notion in real practical examples to help to understand the dynamic behaviour.

However, the concept of the adjustable activity diagram is used in this case study to outline the activities of the traffic subsystem, context acquisition of the traffic service and the drivers (as shown in Figure 6.10.A).

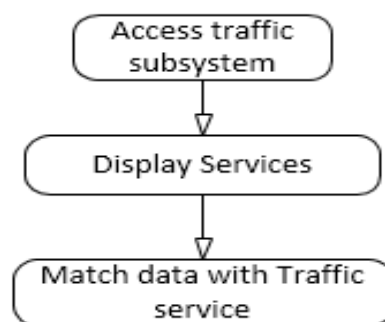


Figure 6.10.A: An example to show the notations of the adjustable activity diagram

Figure 6.10.A used the traditional notations of the adjustable activity diagram to model the behaviour of traffic subsystem activities to display the adaptation services. Also, the state notations are able to represent the operations of the traffic subsystem.

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

- Context Activity Diagram

The most important activities in the traffic subsystem are the adaptation services such as monitoring the movement in such roads and interacting with the navigation subsystem to make real actions, such as suggesting alternative directions and redirecting the driver to the most suitable road to get to their final destination. The concept of the context activity diagram was introduced in Chapter 4 and suggested new notations to demonstrate behaviour activities of adaptation actions and their constraints.

In addition, this concept helps to fulfil the requirements of the traffic subsystem to support the driver's needs at any time and place to produce the necessary services depending on the driver's preferences and needs (as shown in Figure 6.10.B).

The driver specifies user preferences to achieve traffic information using the traffic subsystem, which expresses the activities of traffic services for drivers in different situations during the journey. The traffic subsystem interacts with the traffic service to retrieve the required CIs needed for the drivers, and also suggests the most suitable route for the driver if there is an unexpected event such as a car accident.

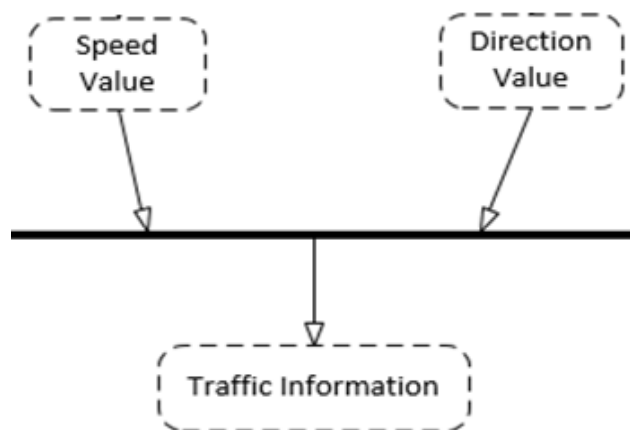


Figure 6.10.B: An example to show the notations of the context activity diagram

- Context Aware Activity Diagram

The context-aware activity diagram notion merges the concepts of the adjustable activity diagram and the context activity diagram and their notations to model the

**CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS**

overall behaviour of any traffic subsystem. This chapter provides a case study of navigation system using the context-aware activity diagram notion to describe the behaviour in models, while navigation system consists of two subsystems: the traffic subsystem interacts with the navigation subsystem to fulfil the driver's needs in terms of traffic situations and road movements. The activities and interactions between the components of the traffic subsystem can be illustrated using a standard modelling of high-level interactions.

This case study also demonstrates the interactions for each component using a standard modelling of low-level interactions to outline the interactions' details (each component in the swimlane and each swimlane describing the component interactions in meta-swimlanes) for the driver, the traffic subsystem and the traffic service (as shown in Figure 6.10.C).

Table 6.6: Summary of behaviour activities for the traffic subsystem

Traffic subsystem	John
Adaptation Action	Traffic information
Acquisition Sense	Traffic service produces the user's direction and speed
CI Type	Atomic CI: direction and speed Composite CI: traffic information
Constraint	None

**CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS**

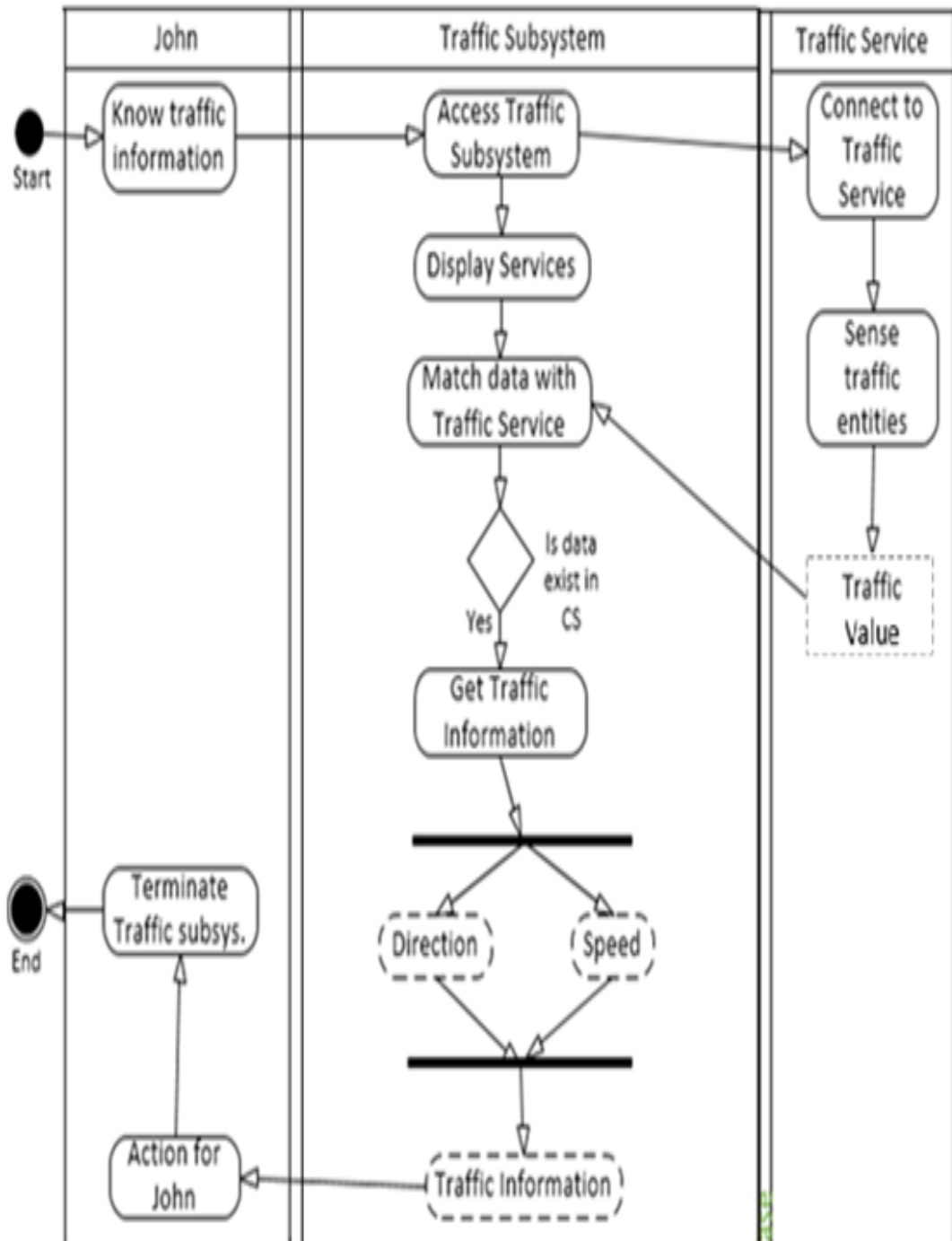


Figure 6.10.C: The main activities of the traffic subsystem

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

6.4.3 Designing the static structure of the Traffic Subsystem using the context-aware class diagram notion

In the case study of navigation system, it is necessary to design the main components of navigation system and their objects and contexts, examining traffic requirements and specifying their functionalities to be developed by different modelling approaches and tools (such as the Tactical Modelling approach and the DoS tool). The tools and modelling concepts of traffic services modelling are still limited in their ability to design the main components. The context-aware class diagram notion suggests a new standard modelling to enhance the structural requirements of CAAs such as traffic subsystems using Microsoft's Visio tool.

- Adjustable Class Diagram

The concept of the adjustable class diagram uses the basic elements of the class diagram to outline the main context objects of the traffic subsystem (as illustrated in Figure 6.11.A) and set the types of class relationships that can be used in this approach. In addition, the most important context object is to know the traffic situation in a given road. This calls for many CIs, such as speed, movement and so on, which are extracted by sensors on the road or via exchanges between CIs and special web services such as traffic services.

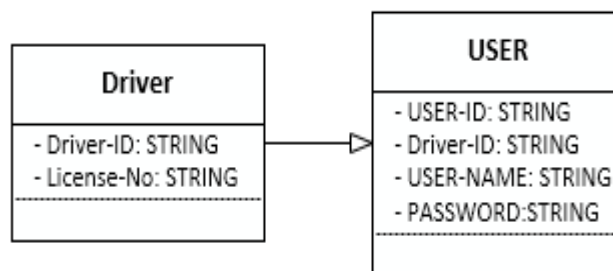


Figure 6.11.A: An example of class diagram elements for the traffic subsystem

The above diagram illustrates an example of an adjustable class diagram using a basic shape to design the user and driver as context objects. The relationship between the

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

driver and the user is <generalization> to identify the <inheritance> relationship between parent and child classes.

- Context Class Diagram

The context class diagram concept proposes new elements, the main one of which is context shape, to express the context properties and operations. When designing the structure components of contexts, it is important to specify the values needed for the classes of the traffic subsystem, such as knowing the remaining time entities of the speed and distance values to complete the functions of the navigation subsystem. In other words, the speed values are captured by the traffic subsystem and the navigation and traffic subsystems are integrated to support each other and work as one system (navigation system).

Providing a reliable estimate of the remaining journey time is the most challenging objective for traffic and navigation subsystems, which need high level strategies to design thorough models that possess the specific requirements of journey durations between different nodes, taking into account the impact factors of roads, junctions and networks and their situations, which affect the final output of remaining time (as depicted in Figure 6.11.B).

**CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS**

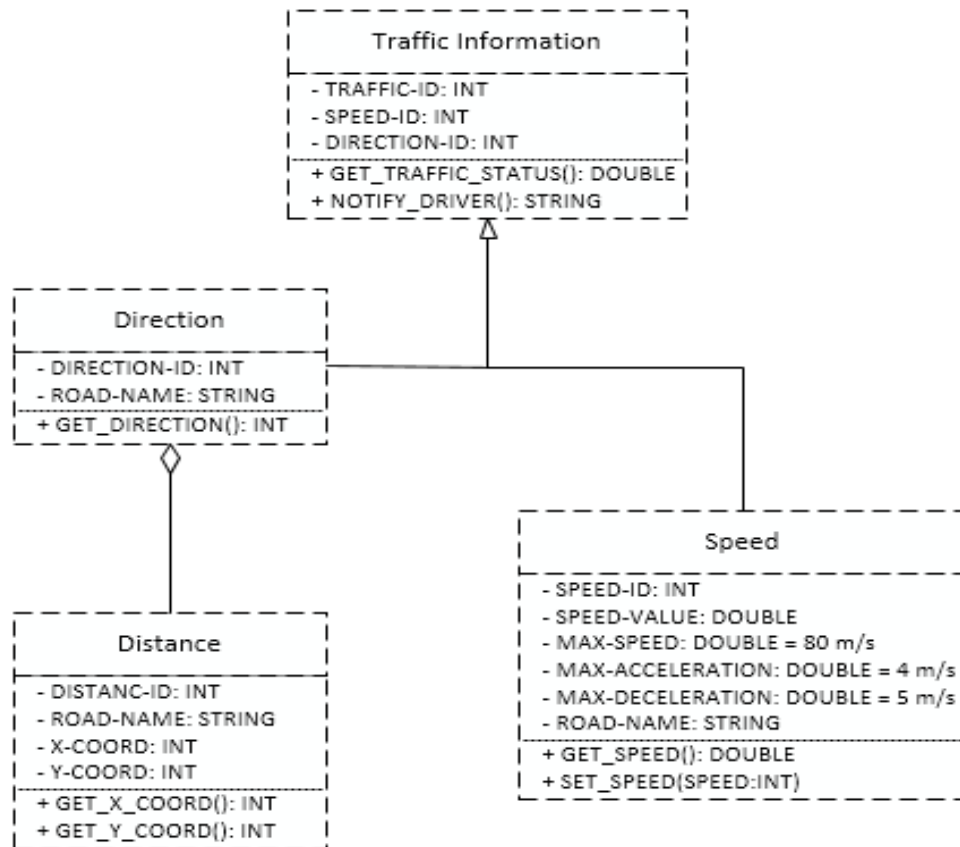


Figure 6.11.B: An example of context class diagram elements for the traffic subsystem

- Context-Aware Class Diagram

The classes and contexts in Figure 6.11.C are designed using the context-aware class diagram notion 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 behaviour 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 behaviour.

However, static and dynamic modelling for run-time of traffic services are not traditional modelling: they need to collect live parameters via special sensors on the roads as well as the performance of traffic services affecting the remaining journey time immediately. The following diagram expresses the context-aware class diagram notion to illustrate the classes and contexts of the traffic subsystem.

**CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM
USING CA-UML DIAGRAMS**

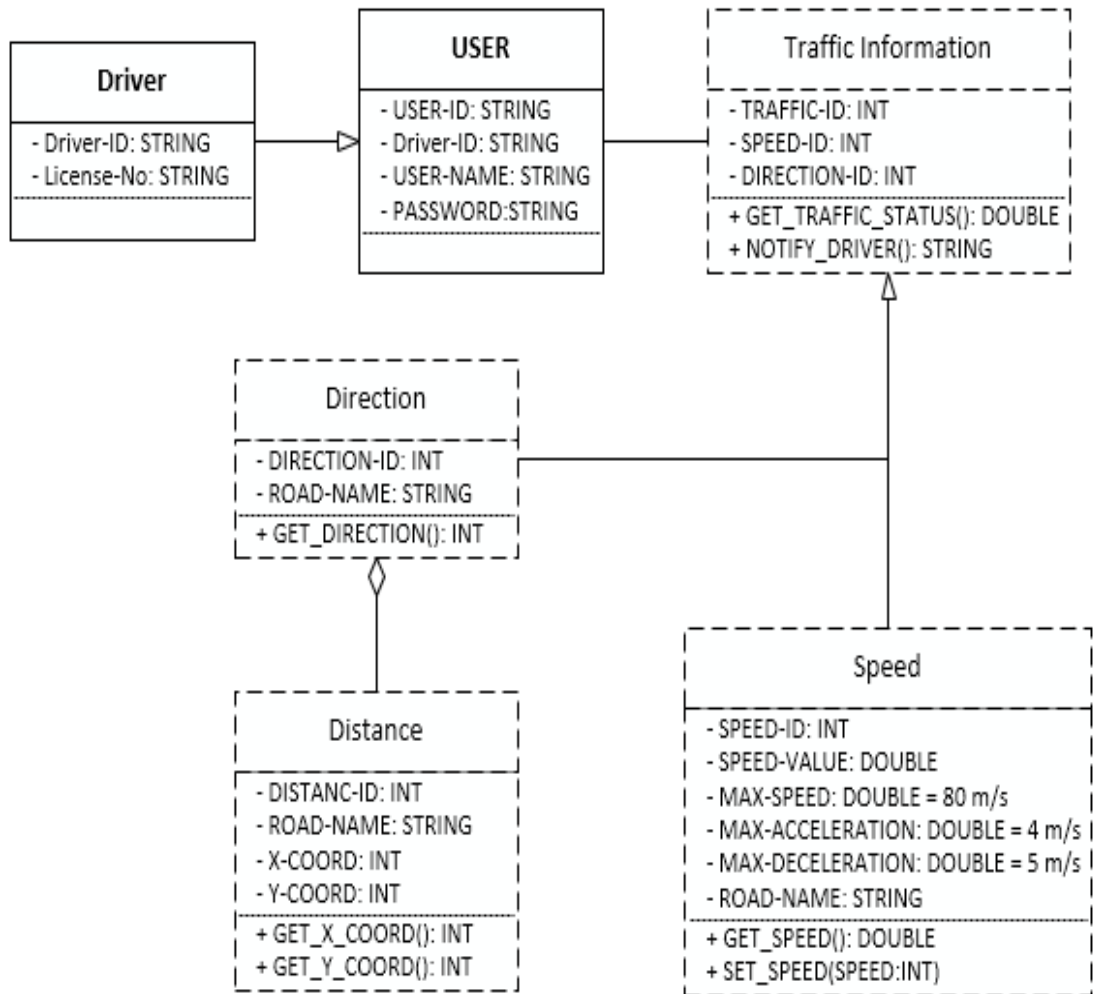


Figure 6.11.C: The static structure of the traffic subsystem

6.5 Summary

This chapter has investigated a real case study and described a range of issues for navigation system and their requirements. It also outlines in clear models and diagrams how CI providers interact with navigation system in real practical studies using an extension of UML use case, activity and class diagrams. This research has used the CA-UML standard modelling of different diagrams to express behaviour and structure in simple diagrams. The case study is about navigation system, which includes two subsystems to depict the behaviour of navigation system and the static structure of their

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

components. However, this case study needs different types of CS to make context acquisition to produce CIs as inputs of parameters to be carried to systems collecting different Atomic CIs relating to each other, which lead to the creation of Composite CIs using special methods; the development results in this case study supported theoretical studies of the CA-UML approach, as explained in previous chapters (Chapters 3, 4 and 5).

In addition, the case study of navigation system studied and depicted the system's main requirements and needs. There are two subsystems working together to provide navigation and traffic services for drivers: a traffic subsystem acquires traffic updates and notifies drivers about traffic on the chosen route, while a navigation subsystem renders maps or retrieves the appropriate map for a specific journey and specifies the destination point with other services, such as known faster routes. These subsystems need many CIs, such as speed, direction, position coordinates, time to destination and map data or details, which are provided by CSs (such as GPS) or web services (such as traffic services).

Finally, navigation system functionalities, behaviour and structure are depicted by the context-aware use case diagram, context-aware activity diagram, and context-aware class diagram, including new notations expressing CIs and CSs; a case study of a navigation system was investigated using three notions as follows:

- A new notion, the context-aware use case diagram, including three concepts of adjustable use case diagram using the traditional notations of the use case diagram to specify the functional requirements of navigation system to show how to depict the functional requirements of navigation system with different usages. Another concept of an extension modelling called the use context diagram is used to show the use of new notations to model the context awareness requirements of the navigation system. Both concepts of the adjustable use case diagram notations and use context diagram notations are merged in the context-aware use case diagram. Moreover, a third concept, called the context-aware usage scenario template, is used to

CHAPTER 6. REAL-LIFE CASE STUDY OF A CONTEXT-AWARE NAVIGATION SYSTEM USING CA-UML DIAGRAMS

describe the functional requirements and context awareness requirements in clear English sentences and scenarios to make the navigation system functionalities clear.

- A new notion, the context-aware activity diagram, including two concepts of the adjustable activity diagram using the traditional notations of the activity diagram to specify the navigation system activities and providers of CIs for this case study and also their interactions and responses. Another concept of extension modelling called the context activity diagram is used to show how new notations can be used to model adaptation actions and their constraints. Both concepts of adjustable activity diagram notations and context activity diagram notations are merged for the context-aware activity diagram notion.
- A new notion of the context-aware class diagram including two concepts of the adjustable class diagram using the traditional elements of the class diagram to design the system structure and its objects for this case study and also to specify the relationships between classes. Another concept of extension modelling called the context class diagram is used to show the use of new elements to design the context structure and specify its entities. The two concepts of adjustable class diagram elements and context class diagram elements are merged in the context-aware class diagram notion.
- This chapter also identified common misunderstandings relating to CA-UML and explained how to use each notation and what it means. It examined CA-UML diagrams in more practical uses. Moreover, this chapter applied the notions of the context-aware use case diagram, the context-aware activity diagram and the context-aware class diagram as well as using their notations and connecting them via different types of relationship in many situations. The following chapter demonstrates a context aware weather forecast system using CA-UML.

7. Real-life Case Study of a Context-Aware Weather Forecast System using CA-UML diagrams

Objectives:

- Evaluate the CA-UML approach in this chapter using a real case study;
- Use new concepts and their notations of CA-UML to model a context-aware weather forecast system;
- Define the user's needs for a context-aware weather forecast system;
- Specify the functional requirements and context-awareness requirements of a weather forecast system using the notion of a context-aware use case diagram;
- Model the dynamic behaviour of a weather forecast system using the notion of a context-aware activity diagram;
- Design the static structure of a weather forecast system using the notion of a context-aware class diagram.

7.1 Introduction

System integration can resolve many problems for systems and applications which might be encountered by traditional or intelligent systems. In other words, not all traditional systems must be context aware [108]. Systems' integration with smart applications to provide awareness of such CI is an alternative solution to capture any CI at any time or place. Weather forecast system may interact with other traditional systems involving basic system data, parameters and results, also specifying the context awareness requirements identified by all CI and CSs when needed. This case study also aims to explain how CAAs and traditional systems acquire CI through interaction with intelligent systems. Examples include civil defence alarm systems and car maintenance systems: civil defence alarm systems alert citizens through a cell phone application by sending alarm messages which contact the GPS to specify the user coordinates; traditional systems can connect to GPS to be context-aware, as in the example of a car

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

maintenance system which automatically contacts a mechanic when an automobile is in need of repair, either alerting a nearby car repair service or giving directions for the location of the stranded vehicle [50, 64, 67].

This chapter presents a real-life case study of a weather forecast system. Mapping and visualization for the context awareness requirements are outlined to ensure that many users have access to the Ubicomp that is relevant to their needs. It also presents context awareness requirements for traffic services and focuses on a weather forecast system that runs on the user's 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 web service, given the limited computational power of a mobile phone.

In addition, this chapter illustrates how the proposed approach of CA-UML diagrams can be used in practice. Traditional UML diagrams are still facing hard issues in terms of modelling and designing smart systems such as weather forecast system. Although existing modelling approaches and tools do not meet all weather forecast system requirements, CAA developers should use a set of concepts and tools to help to execute related elements of CAAs and establish a comprehensive approach for the development life-cycle of CAAs, which is still poor. Sufficient models of CA-UML describe the functionalities, behaviour and structure of CAAs as well as handling their needs in terms of CIs and CSs.

In particular, the results of CA-UML present new extensions modelling of context-aware use case diagrams, context-aware activity diagrams and context-aware class diagrams, merging their new notations and traditional UML notations together with the aim of increasing the quality of models, which will help to decrease the complexity of weather forecast system. Finally, the pragmatics and flexibility of the proposed extensions of UML diagrams are demonstrated using a real-world case study of a weather forecast system using CA-UML diagrams.

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

7.2 The infrastructure of a Context-Aware Weather Forecast System

The main components of weather forecast system infrastructure are similar for different types of smart systems. There are three main components that should be constructed for any smart system that needs to display context-awareness [117]. This chapter also investigates another case study involving a weather forecast system to evaluate CA-UML diagrams in rich examples, express weather forecast system as an example of CAA and show how to apply the CA-UML approach in practical ways. In addition, the two case studies will show clearly how CIs are calculated using the data provided by the corresponding CSs. For example, user preferences control weather forecast system services by indicating what information is stored in the user profile. Weather forecast system services also process CIs and produce actions depending on these preferences. Weather forecast system services can guide the users through different devices, such as smart phones, laptops and PDAs.

The three levels of infrastructure for weather forecast system are shown in Figure 7.1.

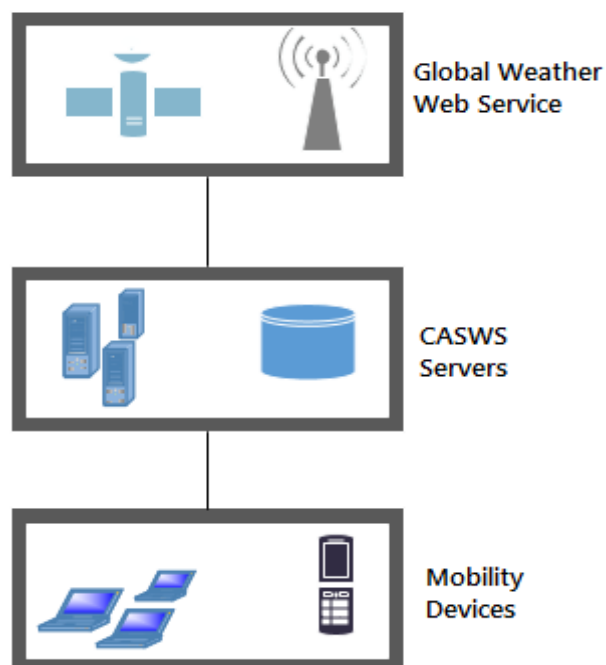


Figure 7.1: Context-aware weather forecast system Infrastructure

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

The weather forecast system is principally comprised of three aspects as follows:

- CI Provider

This element is used to sense the weather conditions using special sensors to read specific entities and carry weather-related CIs to weather forecast system. A smart system such as weather forecast system is aware of the weather changes using set types of CSs such as *WhereAmIService* and *GlobalWeatherWebService* to produce information about weather status and its changes.

- Weather forecast system Server

This element is used to control the weather objects and their parameters, coordinating with weather sensors and services, and also to specify the services needed for users, such as searches for weather locations using GPS. In addition, the objects of weather forecast system set the structure for retrieving information automatically for users depending on the availability of CSs and their CIs.

- Weather forecast system Mobility

In smart systems such as weather forecast system, the user preferences are the most important user requirements, as they inform the weather forecast system server early about the user's needs, but not in the system run-time of weather forecast system. Controlling the services of weather forecast system for users depends on user preferences to be aware of the weather conditions, as well as real life weather changes such as humidity, snow, rain and temperature. In addition, the user information is stored in the mobile phone as preferences such as user location, which are gathered from CSs to be embedded in the mobile phone.

7.3 Modelling a Weather Forecast System using CA-UML diagrams

Ubiquitous computing produces a complex technology providing smart services that respond to the environment and are aware of changes to fulfil the user's needs, such as weather forecast systems. Recently researchers in many institutes have become interested in smart systems for different purposes, such as health, social and educational institutes [93, 114]. CA-UML diagrams investigate the requirements of weather forecast system, including functional requirements and context-awareness requirements, to display the exact weather information on the user's mobile phone depending on the user's location and preferences. At this level, the modelling and designing of weather forecast system specifies what CIs are provided to meet the user's needs.

This section evaluates weather forecast system and outlines its behaviour and structure using CA-UML diagrams which include the context-aware use case diagram, context-aware activity diagram and context-aware class diagram, which are applied in this thesis to visualize, document and structure weather forecast system and express the user preferences in clear models. The user can access information using the weather forecast system; the weather forecast system databases provide information related to the location and other information such as the user's stored location or personal preferences, although such services require interaction with different CSs.

Consider smart systems such as weather forecast system that run on the user's mobile phone and are 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, in addition to information based on mobile communications and computing services. It is assumed that the actual weather data are provided by a weather service, given the limited computational power of a mobile phone. This case study aims to explain how weather forecast system acquires CIs through capturing methods using different CSs. It is suggested that all systems may be context aware. Systems' integrations with CSs provide an alternative solution to awareness of any CI at any time or place.

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

7.3.1 Specifying the functionalities of a Weather Forecast System using the context-aware use case diagram notion

Weather forecast system provides a set of functionalities such as ‘know temperature’, ‘know rain status’, ‘know snow status’, ‘know humidity’ and ‘know location details’. The main functionalities of weather forecast system are to give details of weather conditions for a given location (such as a city, region or country).

This section provides the main functional requirements of weather forecast system using the concept of the adjustable use case diagram through traditional notations of the use case diagram but for different uses to fulfil the functional requirements of weather forecast system. This section also provides the main context awareness requirements of weather forecast system and uses the concept of the use case diagram through new notations of the use context diagram to complete the functional requirements. These functional requirements and context awareness requirements of weather forecast system will also be explained in detail using a context-aware usage scenario template. In addition, this section will explain the functional requirements and context awareness requirements using both a use case diagram and a use context diagram and will depict those requirements together using the notion of the context-aware use case diagram.

- Adjustable Use Case Diagram

Using the existing notations of the use case diagram, it is possible to model the functional requirements of weather forecast system, but each notation is adjusted for other behaviour activities to fulfil weather forecast system requirements.

Modelling starts with the adjustable notations of the use case diagram to show the main functional requirements of weather forecast system. In addition, this case study examines the smart mobility extension with CAAs. The functional requirements of weather forecast system are depicted in this research using an adjustable use case diagram using the same notations as the use case diagram.

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

User preferences specify the user needs in smart systems and also give the user control over the outputs of weather forecast system, such as specifying the country, region, city, preferred language, auto device orientation and user calendar.

- <User preferences> sets the services that are needed for weather forecast system users, such as auto-rotation of the screen that displays the weather forecast system services. <Auto device orientation> is an optional service for users which is a smart hardware behaviour in smart devices such as smart phones which rotate the device screen when the user rotates the device to keep looking at the device screen. <User calendar> calculates the current user activity using information stored in the user calendar; also, the user's location may be inferred from the calendar and provided as an optional CI (as shown in Figure 7.2.A).

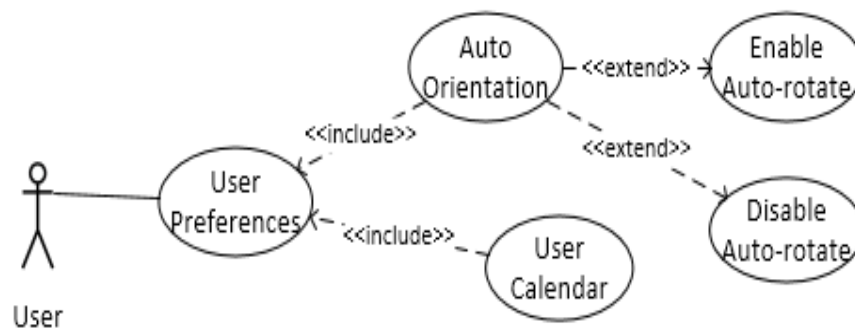


Figure 7.2.A: Functional requirement of user preferences for weather forecast system

- <Display weather forecast> and <User profile> are functional requirements of weather forecast system that include checking weather conditions (composite functions); the basic notation of the use case is able to represent this functional requirement (as shown in Figure 7.2.B), which expresses Composite CI needs as a set of Atomic CIs captured by a different CS.

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

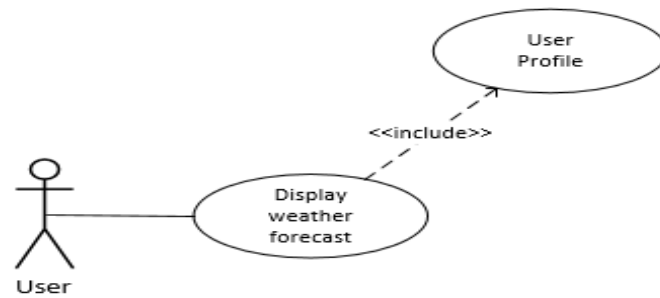


Figure 7.2.B: Functional requirement of display weather forecast for weather forecast system

- Use Context Diagram

To complete the modelling of weather forecast system, context awareness requirements are used to fulfil the functional requirements, such as checking weather conditions, which is the main functional requirement. Weather conditions include different instances of Atomic CIs, such as temperature, rain, snow and so on; in addition, the notations of the use context diagram have the ability to specify the context awareness requirements to fulfil weather forecast system requirements. The context awareness requirements of weather forecast system are snow, location, rain, humidity, nearby location, temperature and auto-orientation as follows:

- <Rain>, <Location> and <Temperature> are mandatory context awareness requirements to notify the user about the weather status in terms of rain and temperature in a given location. <<Include>> relationships between use contexts' notations mean mandatory context awareness requirements for weather forecast system, which outline the relationship between the use context of <weather conditions> and different CIs, represented by use context notations such as <location>, <rain> and <temperature> (as shown in Figure 7.3.A).

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

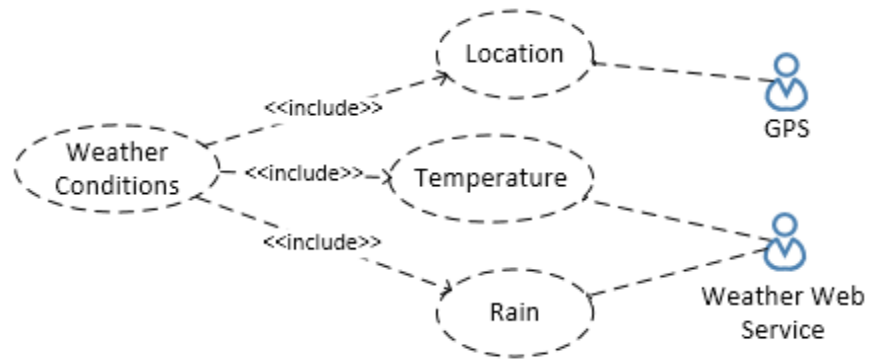


Figure 7.3.A: Context awareness requirements of location, temperature and rain for weather forecast system

- <Snow> and <humidity> are optional services when using weather forecast system, meaning that the user can enable or disable this service. The user also has the ability to find out the percentage humidity. An <<Extend>> relationship is used for optional services such as snow status and humidity (as shown in Figure 7.3.B).

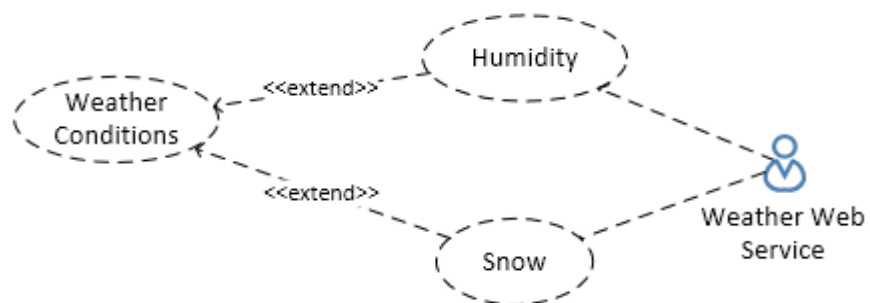


Figure 7.3.B: Context awareness requirements of snow and humidity for weather forecast system

- Context-Aware Use-case Diagram

Consider a case study involving the services of weather forecast system expressing the context-aware use case diagram notion of adjusted and extended notations to demonstrate their effectiveness. To support the proposed notion of the context-aware use case diagram, the case study of weather forecast system shown in the

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

diagrams depicts all weather forecast system functionalities which specify the functional requirements and context awareness requirements. In addition, this modelling of weather forecast system is an example of a CAA run by mobile devices to specify user functional requirements and CAA context-awareness requirements.

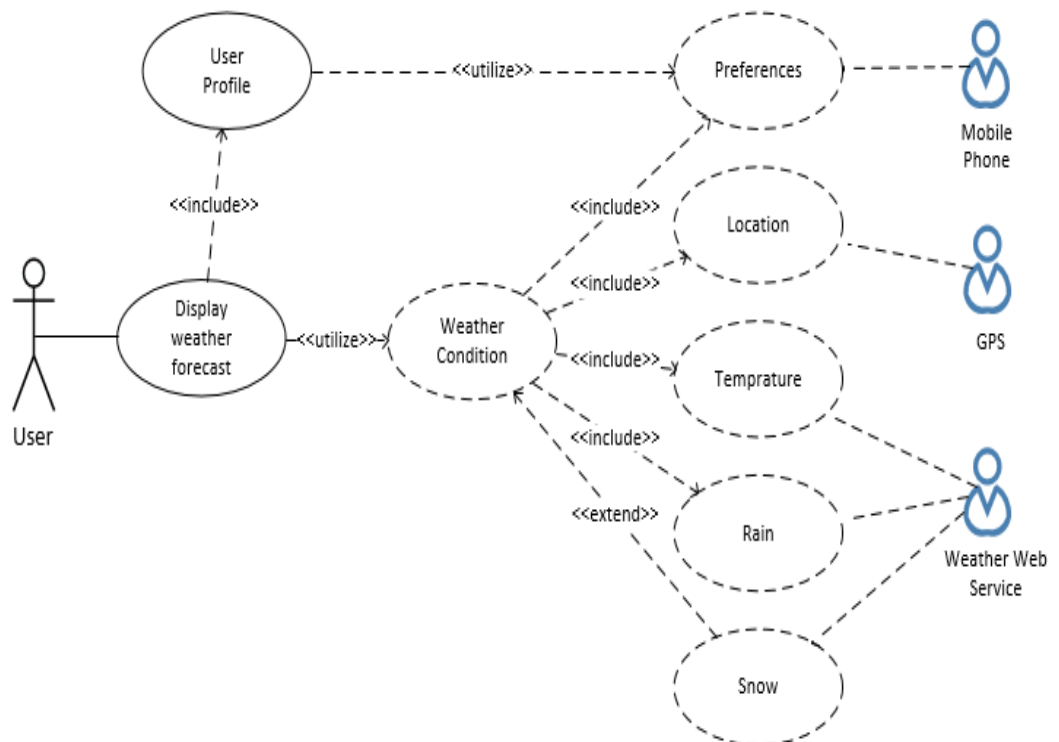


Figure 7.4: A weather forecast system

The main objective for the concept of the adjustable use case diagram is to outline functional requirements for weather forecast system which provides services for users as context awareness. Functional requirements should be aware of CI behaviour and changes. However, the behaviour of CI affects the final actions produced by weather forecast system. A summary of functional requirements and context awareness requirements is shown in Table 7.1, which specifies the overall functionalities of weather forecast system, as well as which CIs are required for weather forecast system, and which CI providers are available.

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

Table 7.1: Summary of functional requirements and context awareness requirements for weather forecast system

Weather forecast system		
Functional requirement	Atomic CI	CI Provider
Check weather conditions	<ul style="list-style-type: none"> - Location - Temperature - Rain status 	<ul style="list-style-type: none"> - GPS - Global weather service
Specify user preferences	<ul style="list-style-type: none"> - Device screen orientation - Preferences 	<ul style="list-style-type: none"> - Embedded Sensor

In addition, the context-aware use case diagram notion provides a set of examples of weather forecast system services, including functional requirements, to display the weather services and a user profile in which it is important to set the user preferences, as provided in Figure 7.4, as mandatory services using the <<include>> relationship. In the meantime, context awareness requirements are also provided, and are divided as follows. Mandatory context awareness requirements are provided using the <<include>> relationship to make important services for the system, such as awareness of location and temperature. Furthermore, snow status and humidity are not so important for the system in hand, but can be provided as optional context awareness requirements using the <<extend>> relationship. Finally, the <<utilize>> relationship retrieves CIs to complete the functional requirements of the weather forecast system.

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

In other words, Figure 7.4 expresses the modelling of the main functional requirements linked with context awareness requirements using the <<utilize>> relationship to depict all functionalities of weather forecast system when visiting a new location. Weather forecast system include context awareness requirements which need a set of Atomic CIs provided by a range of CSs to make information relating to weather conditions available to users at any place or time. By using weather forecast system, the user can find information about weather conditions at his location and be aware immediately of any changes in the weather environment.

Finally, Figure 7.4 illustrates the context-aware use case diagram notion 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 functional requirement of <user preferences> indicates what information stored in the user profile is relevant for this case study. The use case <display weather condition> utilizes the use context <preferences> and the use context <weather condition> to display the weather information according to the user's preferences (such as font size, colours, and so on). The <<include>> relationship is used to indicate the user's location, the current temperature value and the rain status, which are mandatory CIs for this system. However, the snow status and humidity data are optional CIs; hence, the <<extend>> relationship is used for the corresponding use contexts.

- Context-Aware Usage Scenario Template

The modellers need to use the context-aware usage scenario template concept, and it might be mandatory for a thorough description of weather forecast system requirements to be understood by programmers and developers. However, the specific weather forecast system requirements of functional requirements, non-functional requirements and context awareness requirements might be misunderstood in diagrams, and that will cause real problems later in system testing and implementation. In addition, the context-aware usage scenario template concept in this case study aims to define many scenarios for weather forecast system services by writing a story of system scope that controls the

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

conditions between users, weather forecast system and CI providers. Moreover, the context-aware usage scenario template helps to move from context-aware use case diagrams to the following notion of context-aware activity diagrams in clear words.

Table 7.2: Functional requirement and context awareness requirement of weather forecast system using context-aware usage scenario template

Elements	Example
Use Case Name	<ul style="list-style-type: none"> - User preferences - Auto device orientation - User calendar
Use Context Name	<ul style="list-style-type: none"> - Device screen orientation - User activity
Use Case ID	<ul style="list-style-type: none"> - User preferences: 55 - Auto device orientation: 56 - User calendar: 57
Use Context ID	<ul style="list-style-type: none"> - Device screen orientation: X30 - User activity: H25

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

<p>Context-aware use case diagram</p>	<pre> graph TD User((User)) --- UP((User Preferences)) UP --<<include>> AO((Auto Orientation)) AO -.-> <<extend>> EA((Enable Auto-rotate)) AO -.-> <<extend>> DA((Disable Auto-rotate)) EA --<<utilize>> DSO((Device Screen Orientation)) DSO -.-> ES((Embedded Sensor)) UC((User Calendar)) --<<include>> UP UC -.-> <<utilize>> UA((User Activity)) UA -.-> CA((Calendar)) </pre>
<p>Use Case Priority</p>	<p>Low</p>
<p>Use Context Priority</p>	<p>Low</p>
<p>Use Case-55 Scenario narrative</p>	<ul style="list-style-type: none"> - The user accesses the weather forecast system interface - Weather forecast system will display the main screen - The user has to insert preferences such as preferred language and location - If the user inserts preferences - Weather forecast system retrieves weather information about the selected location in the preferred user language

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

	<ul style="list-style-type: none"> - The user can select other optional services such as auto device orientation - Else weather forecast system will choose English language and the current location of the user as the default setting
Use Case-56 Scenario narrative	<ul style="list-style-type: none"> - The user has an optional service to enable orientation - If the user sets auto device orientation - Enabling this service will rotate the screen of the user device when moved to another direction - Else user can keep this service disabled as default
Use Case-57 Scenario narrative	<ul style="list-style-type: none"> - The user can select an optional service of user calendar - If the user enables this user calendar service - Weather forecast system will calculate the user's current activity using information stored in the calendar - Else user can keep this service disabled as default
Use Context- X30 Scenario narrative	<ul style="list-style-type: none"> - The device screen orientation interacts with user behaviour - The device orientation is captured using special sensors embedded in the user's device - This context awareness requirement is important for smart devices and their applications and systems, such as weather forecast system
	<ul style="list-style-type: none"> - The user device has the ability to store the user's location

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

Use Context- H25 Scenario narrative	<ul style="list-style-type: none"> - The user calendar can be captured through sensors - Weather forecast system will calculate the user's activities using special sensors embedded in the device
Actor	User
Use Case-55 conditions	<ul style="list-style-type: none"> - User has to insert the location and preferred language - Else weather forecast system will display the service and retrieve weather information about the user's current location in English
Use Case-56 conditions	None (optional service of auto device orientation)
Use Case-57 conditions	None (optional service to know user's calendar)
Use Context- X30 Constraints	<ul style="list-style-type: none"> - If the user keeps the service of auto device orientation disabled - No action - Else enable weather forecast system to do service of auto device orientation to rotate with user movements
Use Context- H25 Constraints	<ul style="list-style-type: none"> - If the user keeps the service of user calendar disabled - No action - Else enable weather forecast system to access user calendar and calculate user activities

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

Frequency	Multiple uses per visit
Awareness Requirements	Update user of future events during weather forecast system running such as increasing wind speed or rain stopping
Context Source Name	- Embedded sensor - Calendar
Alternative Context Source Name	None

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

7.3.2 Modelling the dynamic behaviour of a Weather Forecast System using the context-aware activity diagram notion

Recently, mobile requirements have been extended for many uses beyond making calls or text messaging. The uses of smart phones have also been extended to internet surfing, navigation and other context-aware activities.

Weather forecast systems can interact with other systems and may use traditional or intelligent means to find the needed CIs at any time or location. It is also necessary for other external systems to interact with navigation system. The services of the weather forecast system may be extended to include additional outputs such as wind speed and humidity. Furthermore, there are many systems that are unable to interact or communicate with CSs, web services and sensors; also, interaction may not occur with main system functions but occasionally with CSs.

- Adjustable Activity Diagram

Weather forecast system behaviour is always changing because the weather situation changes every day, hour and minute and may change in seconds, such as wind speed or percentage humidity. However, weather forecast system activities can be depicted using traditional activity diagram notations to specify the sequence of flows and activities for weather forecast system behaviour. This case study will investigate the uses of different concepts (such as the adjustable activity diagram) for smart systems such as weather forecast system. Using the same notations that were previously used in the traditional activity diagram, it is possible to describe the activities between weather forecast system and CI providers (as shown in Figure 7.5.A).

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

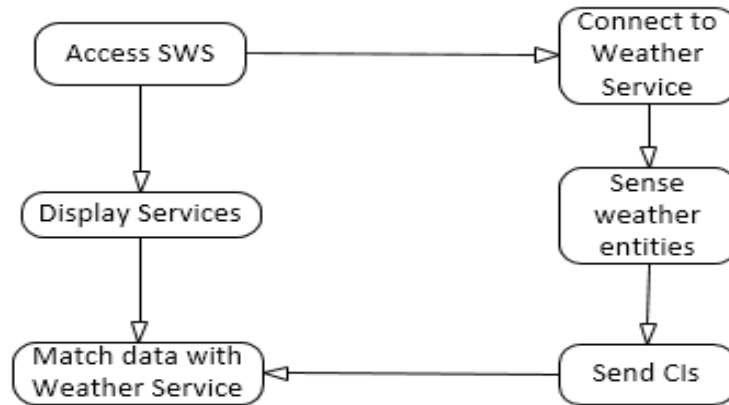


Figure 7.5.A: An example of weather forecast system and CS activities using an activity diagram

The above diagram expresses the activities of weather forecast system and the weather service, and also the interactions between them, using the traditional activity diagram notations.

- Context Activity Diagram

The dynamic behaviour of weather forecast system is challenging for designers of smart systems, which need to capture ever-changing weather behaviour. Numerous sensors are needed to guarantee monitoring and updating. However, the new concept of the context activity diagram helps to specify the activities of context acquisition and CIs that are needed for weather forecast system to make adaptation decisions for users without delays. This concept also provides new notations expressing the adaptation actions and their constraints to control the final output for users. In addition, this research enhances the modelling stage to fulfil the users' needs and outline the affected activities to produce suitable services for users (as shown in Figure 7.5.B).

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

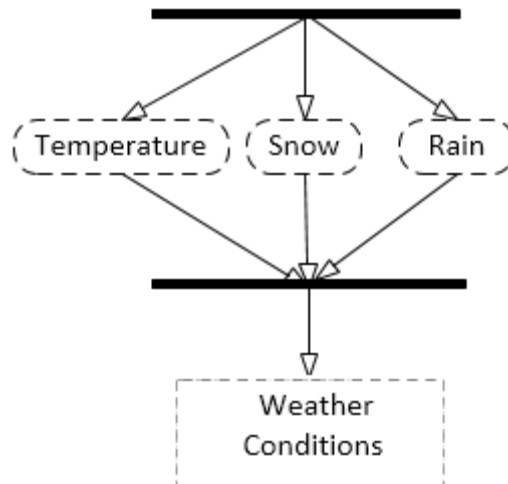


Figure 7.5.B: An example of adaptation actions for weather forecast system using context activity diagram

- Context Aware Activity Diagram

This case study supports the notion of the context-aware activity diagram using the notations of two concepts of the adjustable activity diagram and the context activity diagram to explain the practical usage of the notion of the context-aware activity diagram and its notations. The interaction activities between weather forecast system components are depicted in this case study to classify the main components of the user, the weather forecast system and CI providers using a standard modelling of high-level interactions which create overall behaviour modelling for weather forecast system. The modelling separation between weather forecast system categories helps to understand weather forecast system behaviour and the main factors affect the final actions for users (as shown in the following diagram).

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

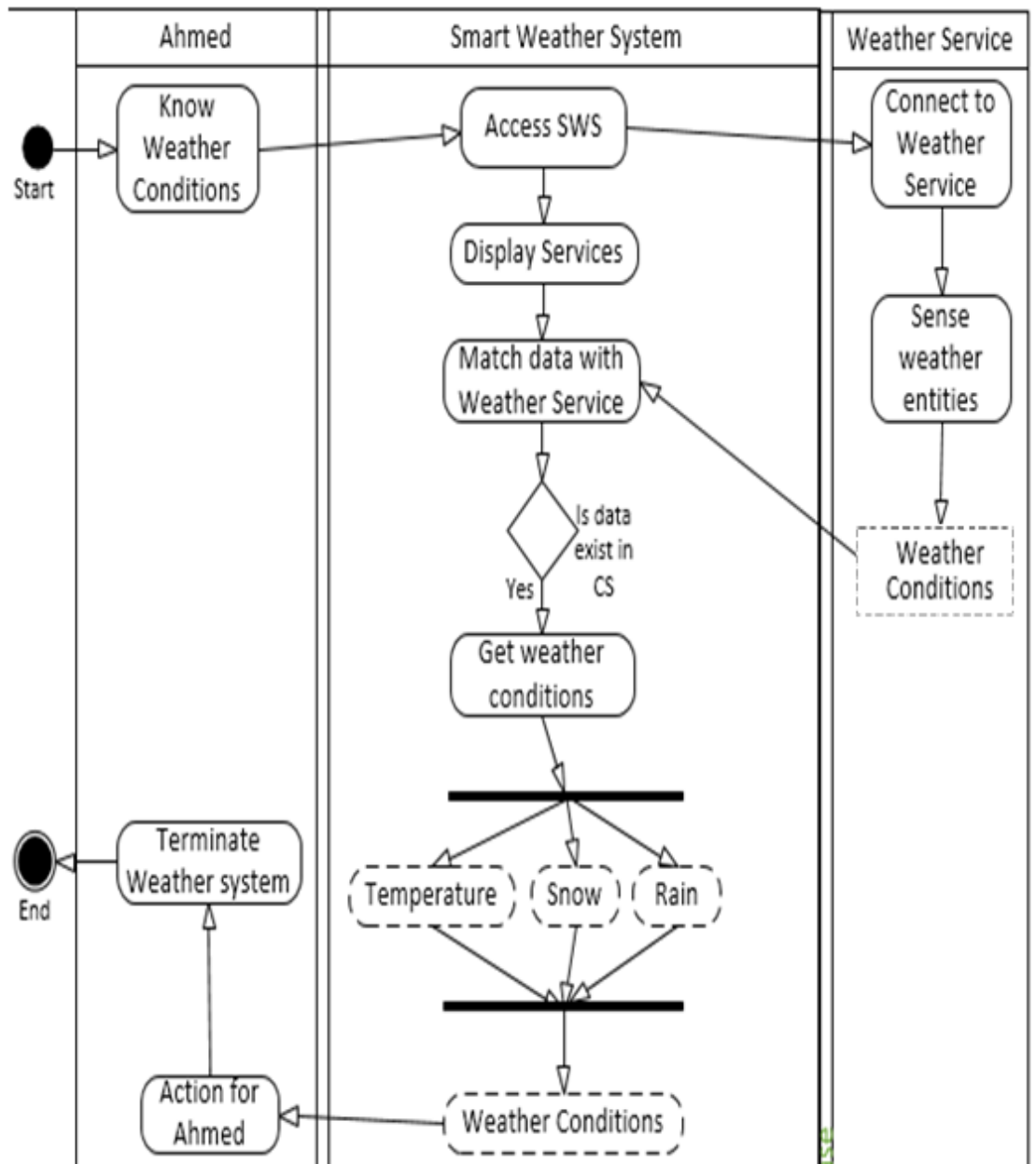


Figure 7.5.C: Weather forecast system behaviour modelling using a context-aware activity diagram

The above diagram uses a standard modelling of high-level interactions to model the dynamic behaviour; the activities of weather forecast system and weather services depicted by activity diagram notations and context activity diagram notations are used to express the adaptation decisions and their constraints to specify the CIs needed to produce weather conditions such as snow, temperature and rain.

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

7.3.3 Designing the static structure of a Weather Forecast System using the notion of the context-aware class diagram

To support the notion of extension of the context-aware class diagram, this chapter provides a case study of a weather forecast system using the elements of both the adjustable class diagram and the context class diagram. This case study evaluates the notion of the context-aware class diagram and sets out to design the main components of the weather forecast system to express the effectiveness of new elements of the CA-UML approach. In addition, this case study 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, weather forecast system services 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, and GPS can be used to gather the user's location. This example aims to explain how designing weather forecast system, specific CIs are needed for the design stage through capturing functions set by different CSs.

- Adjustable Class Diagram

The concept of the adjustable class diagram has the ability to design the structure of weather forecast system objects using traditional elements of the class diagram. In smart systems, the user's preferences and location control the services and final actions for users. The users can find out about the weather conditions as real life weather. However, this component 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 (as shown in Figure 7.6.A).

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

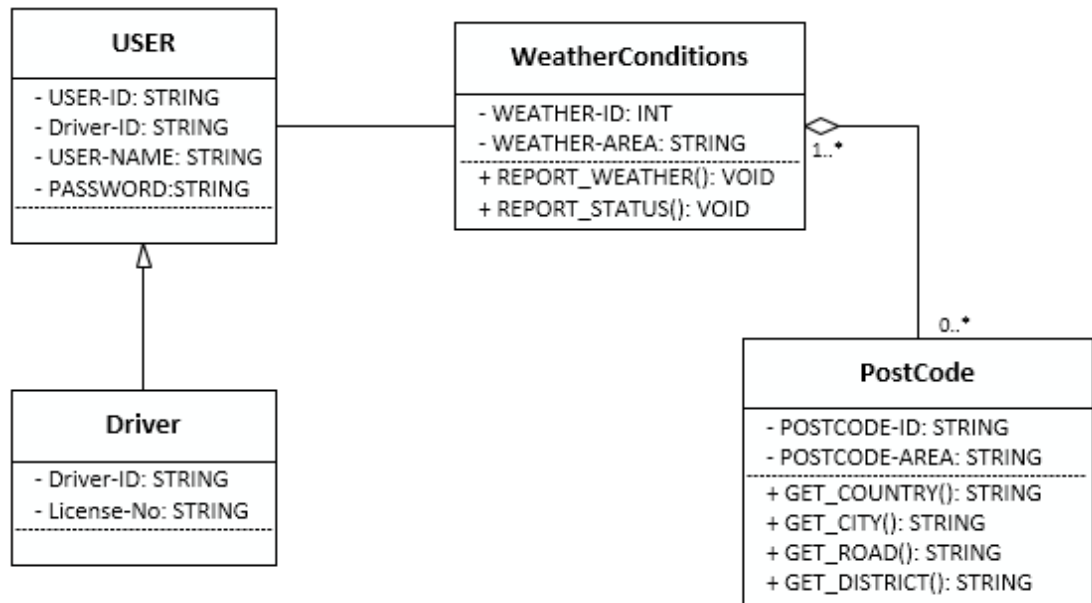


Figure 7.6.A: An example of the weather forecast system using an adjustable class diagram

The above diagram used basic elements of the class diagram and their relationships to design the structure components, and also the class shape to outline all context objects of the weather forecast system and their properties and operations.

- Context Class Diagram

New elements have been identified for a new concept of the context class diagram to express the structure components of contexts and their entities. The context service collects the parameters (context information and its values) for the CAA. These parameters may be atomic CI or composite CI. In addition, CS provides automatic parameters for weather forecast system to produce the CIs required to fulfil the context objects (weather forecast system should receive specific CIs to be executed). The context-aware class diagram notion 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, such as GPS, Global Weather, Weather Web Service and so on (as shown in Figure 7.6.B).

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

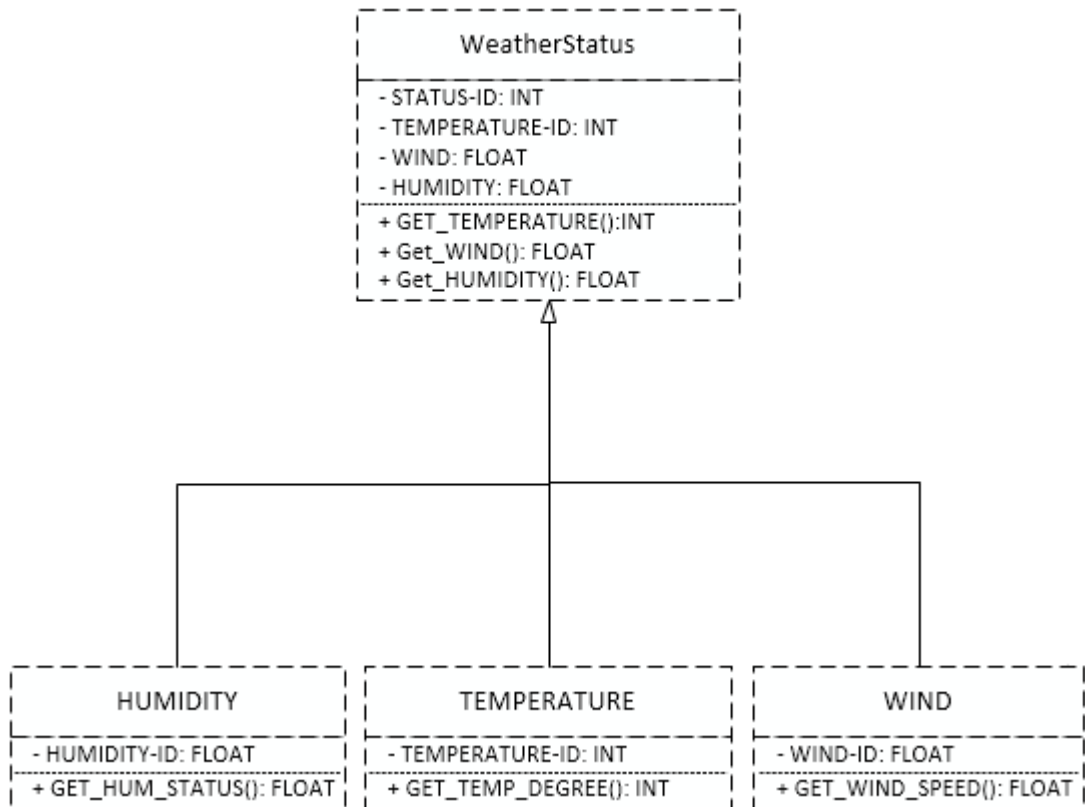


Figure 7.6.B: An example of the weather forecast system using a context class diagram

- Context-Aware Class Diagram

Typically, the structure of weather forecast system consists of three components that are merged into one component that appears in the user profile to give powerful performance to the mobility services that hide the backbone of the components for intelligent systems such as weather forecast system.

The component structure of the weather forecast system outlines the main objects and contexts of weather which specify the services needed for users, such as the search engine for the location. The objects of the weather forecast system set the structure for retrieving information automatically for users depending on the availability of CSs and their CIs. In addition, the component structure of the weather forecast system specifies which parameters should be entered into the system; then the weather forecast system will interact with suitable CS to retrieve the requested information as system parameters for specific functions to produce an action for the user.

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

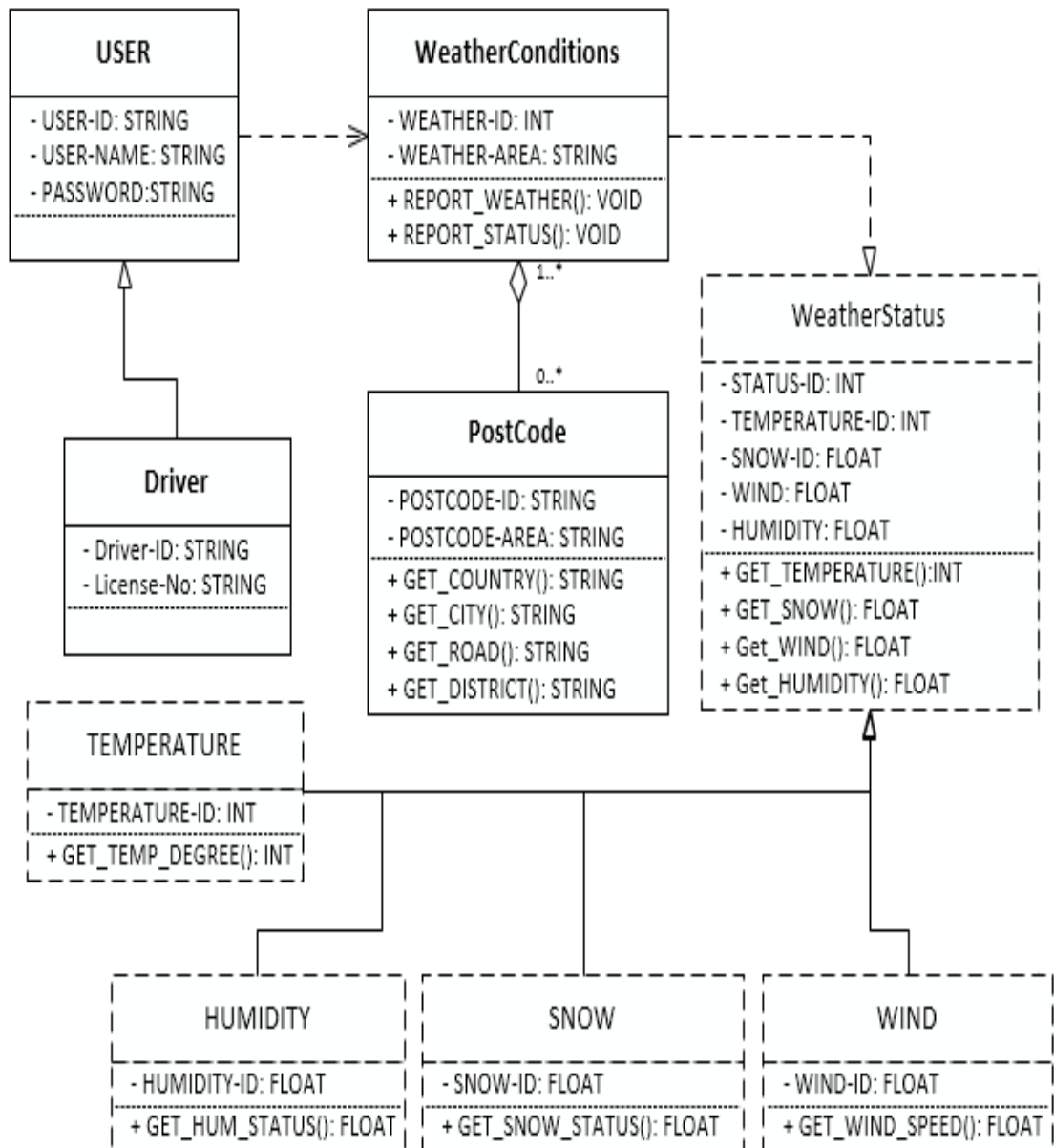


Figure 7.6.C: The structure components of the weather forecast system

The above diagram designs the static structure of the weather forecast system and their components using the basic and new elements of the adjustable class diagram and the context class diagram. This approach simplifies the design stage of the weather forecast system to specify the context objects and context values with each other in one diagram, and also uses a new relationship of utilization, which fulfils the requirements of context objects to retrieve the entities and parameters needed to provide the context awareness requirements for users.

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

7.4 Summary

This chapter has investigated real case studies and described a range of issues for navigation system and weather forecast system and their requirements. It also outlines in clear models and diagrams how CI providers interact with navigation system and weather forecast system in real practical studies using an extension of UML use case, activity and class diagrams. This research has used the CA-UML diagrams to express navigation system and weather forecast system behaviour and structure in simple diagrams. The first case study is about navigation system, which includes two subsystems to depict the behaviour of navigation system and the static structure of their components. However, this case study needs different types of CS to make context acquisition to produce CIs as inputs of parameters to be carried to systems collecting different Atomic CIs relating to each other, which lead to the creation of Composite CIs using special methods; the development results in this case study supported theoretical studies of the CA-UML approach, which have been explained in previous chapters (Chapters 3, 4 and 5).

In addition, this case study models the dynamic behaviour and structure. Weather forecast systems produce different types of CIs which are provided by many CSs to support the users in different locations at any time. There are many traditional systems that are unable to interact or communicate with CSs; also, interaction may not occur with main system services but occasionally with CSs. This case study depicted and designed the main services of weather forecast system which may interact with other CAAs or traditional systems involving basic system data, parameters and results. Finally, weather forecast system functionalities, behaviour and structure are depicted by a context-aware use case diagram, a context-aware activity diagram, and a context-aware class diagram, including new notations expressing CIs and CSs; a case study of a weather forecast system was investigated using three notions as follows:

- A new notion of a context-aware use case diagram including three concepts of the adjustable use case diagram using the traditional notations of the use case diagram to specify the functional requirements of a weather forecast system to

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

show how to depict these functional requirements with different usages. Another concept of an extension modelling called the use context diagram is used to show the use of new notations to model the context awareness requirements of the weather forecast system. The concepts of both the adjustable use case diagram notations and the use context diagram notations are merged in the notion of the context-aware use case diagram. Moreover, a third concept, called the context-aware usage scenario template, is used to describe the functional requirements and context awareness requirements in clear English sentences and scenarios to make the system functionalities clear.

- A new notion of the context-aware activity diagram, including two concepts of the adjustable activity diagram using the traditional notations of the activity diagram to specify the weather forecast system activities and the providers of CIs for weather forecast system and also their interactions and responses. Another concept of extension modelling, called the context activity diagram, is used to show how new notations can be used to model adaptation actions and their constraints. The concepts of both adjustable activity diagram notations and context activity diagram notations are merged for the notion of the context-aware activity diagram.

- A new notion of the context-aware class diagram including two concepts of the adjustable class diagram using the traditional elements of the class diagram to design the weather forecast system structure and its objects and also to specify the relationships between classes. Another concept of extension modelling, called the context class diagram, is used to show the use of new elements to design the context structure and specify its entities. The two concepts of adjustable class diagram elements and context class diagram elements are merged in the context-aware class diagram notion.

In addition, the weather forecast system is real life case study which enables the designers and developers to understand the differences between CA-UML diagrams and to ascertain when each diagram should be used and why. This

CHAPTER 7. REAL-LIFE CASE STUDY OF CONTEXT-AWARE WEATHER FORECAST SYSTEM USING CA-UML DIAGRAMS

chapter also identified common misunderstandings relating to the CA-UML and explained how to use each notation and what it means. It examined CA-UML diagrams in more practical uses. Moreover, this chapter applied the context-aware use case diagram, context-aware activity diagram and context-aware class diagram as well as using their notations and connecting them via different types of relationship in many situations.

8. Conclusion and Future Work

Objectives:

- Summarize the work in this thesis;
- Give the statement of evaluation;
- List the main contributions of this work;
- Revise the research questions;
- Present the future work.

8.1 Research Summary

This research has defined the context-awareness requirements of CAAs and outlined the context values needed to make self-adaptive services in response to context changes. This thesis has suggested a new approach to extend basic UML diagrams, called CA-UML diagrams, to model the functionalities, behaviour and structure of CAAs. The extension approach of CA-UML also provides clear standard modelling for each diagram used to cater for the specification, visualisation, construction and documentation of CAAs.

More important is the modelling stage, which illustrates the important properties of such applications. CA-UML diagrams have the ability to model and design functional requirements, context awareness requirements, activities and adaptation decisions of CAAs' behaviour and address the stakeholders, functions, properties, objects, and context entities of CAAs' structure. New notations and relationships are established for each notion to express overall maps of CAAs' requirements and their CIs and CSs. In this research, each notion has used new notations to specify CIs and their corresponding CSs to complete the basic modelling of UML. This separation of concerns between adjustable UML notations and new notations can be usefully merged for CA-UML diagrams, especially when dealing with complex smart systems.

CHAPTER 8. CONCLUSION AND FUTURE WORK

In addition, the thesis has described CA-UML and its concepts to specify the main parts of CAAs, which consist of three main notions: firstly, the context-aware use case diagram notion, which is able to model the functional requirements and context awareness requirements of CAAs using three concepts of the adjustable use case diagram, use context diagram and context-aware usage scenario template; secondly, the notion of the context-aware activity diagram, which is able to model the behaviour requirements of CAAs, such as activities, context objects, adaptation actions and their constraints, using two concepts, namely the adjustable activity diagram and the context activity diagram; thirdly, the context-aware class diagram notion, which is able to design the structure requirements of CAAs, such as CAAs' objects as classes and context entities as parameters, using the concepts of the adjustable class diagram and the context class diagram.

This thesis has proposed a novel modelling approach called CA-UML for CAA modelling and design. CA-UML diagrams have been proposed as suitable visualization notations to model the context awareness requirements. Therefore, in this thesis, the UML diagram has been extended into three notions called the context-aware use case diagram, the context-aware activity diagram and the context-aware class diagram in order to enable them to capture the behaviour and structure modelling of CAAs and their requirements.

A summary of this work is reported in this thesis as follows: the thesis has introduced the research motivation, scope, and questions, identified the measure of success and outlined the thesis structure. It then provided a literature review, which has investigated the importance of three fields working with each other to increase the quality and performance of CAAs services: the main topics covered in this chapter are RE, UML and CAAs. It also presented the background of these topics and described different approaches, concepts and techniques for each topic, such as requirements engineering approaches and techniques, UML diagrams and concepts and CAAs' structures and behaviour. Moreover, it has provided an overview of why requirements engineering and UML are important for CAAs' modelling, design and simulation.

CHAPTER 8. CONCLUSION AND FUTURE WORK

Then, the thesis provided a novel notation for this research, which extended the use case diagram by proposing an extension notion known as the context-aware use case diagram to illustrate the functionalities of CAAs using the existing notations of the use case diagram and the new notations of the use context diagram to fulfil the functional requirements and context awareness requirements involving CIs and CSs. The context-aware use case diagram notion enhances the dynamic modelling of CAAs to deliver specific requirements for CAAs, while the context awareness requirements are specified using new notations. Two new concepts – the adjustable use case diagram and the use context diagram – are merged for this notion to capture what CIs and CSs are needed for the functional modelling of CAAs.

The context-aware use case diagram notion suggested a new relationship called <<utilize>>, which links the notations of the adjustable use case diagram and the use context diagram.

Then, the thesis provided a novel notion which extended the activity diagram by proposing an extension notion known as the context-aware activity diagram using two adjustable levels to depict the dynamic behaviour of CAAs using the existing notations of the activity diagram and the new notations of the context activity diagram, and also suggested a new swimlane for CS activities. The notion of the context-aware activity diagram outlines a powerful language for describing the functions and behaviour of CAAs. Two new concepts, namely the adjustable activity diagram and the context activity diagram, are merged for this notion to demonstrate the interactions between CAAs and CSs using swimlanes and their meta-swimlanes to control CAA behaviour as well as to specify adaptation actions and their constraints to be performed in reaction to changes in the application's environment.

Then, this research provided a novel notion which extended the class diagram by proposing an extension notion known as the context-aware class diagram to design the structure models of CAAs using existing elements of the class diagram to enhance the general structure of CAAs and their components. Two new concepts – the adjustable class diagram and the context class diagram – are merged for this notion to depict all context awareness requirements to fulfil the modelling needs of CAAs, designing

CHAPTER 8. CONCLUSION AND FUTURE WORK

contexts in a way that helps to clarify their properties and operations as well as the calculated values of contexts for CAAs' classes to create awareness of the changeable values of contexts which are always affected by environmental factors. The context-aware class diagram notion suggested a new relationship called *utilization* to connect the elements of the adjustable class diagram and the context class diagram.

Finally, the thesis has evaluated the pragmatics of the proposed approach using real-world case studies of a navigation system involving CA-UML diagrams. This chapter has also applied all CA-UML notions and concepts which are introduced in previous chapters using the navigation system as a case study and investigated their requirements and needs. The context-aware use case diagram, context-aware activity diagram and context-aware class diagram are used to express the practical usages of CA-UML (Chapter 6). The thesis has also evaluated the pragmatics of the proposed approach using another real-world case study of weather forecast system involving CA-UML diagrams (Chapter 7). This chapter has also applied CA-UML diagrams, which were introduced in previous chapters, using a weather forecast system as a case study, and investigated their requirements and needs. The context-aware use case diagram, context-aware activity diagram and context-aware class diagram are used to express the practical usages of CA-UML. In addition, this thesis has investigated the modelling, design and simulation of CAA behaviour and structure using a new approach of CA-UML, which has extended basic UML to depict CAAs. This thesis has focused on the modelling and simulation of CAAs using CA-UML diagrams. CA-UML diagrams are suitable for CAAs, providing a set of notions and their concepts that are able to improve CAAs and their environments for different uses, such as defining and developing new context objects and existing objects, and also checking the CAA objects' situation when an object is used for adaptation services and specifying the objects' relationships. CAA is very complicated and needs a professional approach for all steps, and this thesis has focused on the CAAs' modelling and simulation using a new approach of CA-UML.

Moreover, this thesis has explained a set of uses for new context-related notations in UML modelling for CAAs, using the special cases of requirements engineering and modelling by CA-UML diagrams. According to the changes of CI for CAA, the running of the requirements engineering phase takes into consideration such changes

CHAPTER 8. CONCLUSION AND FUTURE WORK

in requirements, while the modelling of those requirements using the UML notations does not sufficiently express these changes in requirements. A use case diagram is a high-level modelling of information systems. This thesis approached the use of use case diagrams with new notations and considered the need to create a related approach of activity diagrams to show low level modelling for CAAs in more detail and with more conditions. However, the activity diagram approach needs to be extended to include new notations that enhance the modelling of CAAs; it is necessary for the activity diagram to explain more details and conditions for CAAs. Finally, the class diagram is a standard design language to model related classes of systems' objects and this research has provided more elements to design the contexts' structure and their entities and parameters.

8.2 Statement of Evaluation

CAA is a new generation of IT and its potential to make life easier is of interest in different fields around the world. UML is a standard modelling language, which includes several diagrams to depict systems' structure and behaviour. UML is also used to define the functional and non-functional requirements of such systems or applications. UML has been extended in this research to model the dynamic behaviour and static structure of CAAs. An extension notion has been introduced for each UML diagram – the use case diagram, the activity diagram and the class diagram.

RE and CAAs' areas are investigated in this thesis using extension UML diagrams called CA-UML diagrams. CA-UML diagrams enhance the modelling and design stages of CAAs to specify the requirements of CAAs' functionalities, behaviour and structure.

To the best of the author's knowledge of UML diagrams, there is no comprehensive research to model context-awareness requirements by use case diagrams, activity diagrams and class diagrams. The existing studies of UML to model CAAs are still limited and insufficient to express CAAs' adaptation actions, CSs' context acquisition, CIs and their changes.

CHAPTER 8. CONCLUSION AND FUTURE WORK

In addition, both functional and context-awareness requirements are depicted to present the functionalities, behaviour and structure of CAAs' requirements and needs.

The research approach of CA-UML diagrams extends the limitations of existing approaches of UML modelling for CAAs.

The proposed extensions have added the following three notions:

- Enabling the use case diagram – a new notion to define the functional requirements and context-awareness requirements for CAAs. This notion, called the context-aware use case diagram, includes two concepts: the adjustable use case diagram and the use context diagram. Both concepts' notations are merged for the context-aware use case diagram notion to model the functionalities of CAAs.
- Enabling the activity diagram: a new notion to define the CAAs' activities, CSs' activities, context acquisition, adaptation actions and their constraints. This notion, called the context-aware activity diagram, includes two concepts: the adjustable activity diagram and the context activity diagram. The two concepts' notations are merged for the context-aware activity diagram notion to model the dynamic behaviour requirements of CAAs.
- Enabling the class diagram, which is a new notion to define the component structure of classes, contexts and their relationships. This notion is called the context-aware class diagram and includes two concepts: the adjustable class diagram and the context class diagram. The two concepts' elements are merged for the context-aware class diagram notion to design the static structure requirements of CAAs.

Two real life case studies have been investigated in this thesis. Navigation and weather forecast systems were modelled and designed based on our proposed extensions to the use case diagram, activity diagram and class diagram, and were presented in detail, including particular scenarios and practical examples to support the theoretical aspects of this research.

CHAPTER 8. CONCLUSION AND FUTURE WORK

8.3 Research Questions Revisited

The work presented in this thesis is evaluated by answering the research questions that were formulated in Chapter 1 to be revisited in this section.

The main research question was: How can we use UML extensions to model the functionalities, behaviour, and structure of CAAs?

However, this question poses many sub-questions as follows:

Q. 1 Can the existing use case diagram notations be extended to model context awareness requirements of CAAs?

The above sub-question was answered in full, with descriptions and diagrams, in Chapter 3 by means of the context-aware use case diagram notion, which extended the use case diagram to model the CAAs' functionalities.

In relation to another target of the activity diagram, another sub-question was developed as follows:

Q. 2 Can the existing activity diagram notations be extended to depict the dynamic behaviour of CAAs?

The above sub-question was answered in full, with descriptions and diagrams, in Chapter 4 by means of the context-aware activity diagram notion, which extended the activity diagram to depict the CAAs' behaviour.

In relation to another target of class diagram, another sub-question was developed as follows:

Q. 3 Can the existing class diagram elements be extended to design the static structure of CAAs?

The above sub-question was answered in full, with descriptions and diagrams, in Chapter 5 by means of the context-aware class diagram notion, which extended the class diagram to design the CAAs' structure.

Finally, a concluding question addressed the practicality of the proposed extensions:

CHAPTER 8. CONCLUSION AND FUTURE WORK

Q. 4 Are the proposed extensions of CA-UML practically applicable to real-life case studies?

The above sub-question was answered in full, with descriptions and diagrams, in Chapters 6 and 7, which presented case studies in which these extensions were practically applied to model and design navigation system and weather forecast system.

In summary, the proposed extensions derived from a deep and synergistic understanding of three aspects: firstly, requirements engineering approaches and their techniques for the life-cycle development of CAAs; secondly, the nature of CAAs and their CIs to cater for context awareness requirements and provide adaptation services for users; thirdly, the power and limitations of existing UML diagrams.

8.4 Contribution to Knowledge

In this research, the existing notations of the use case diagram have been extended to model functional requirements and context awareness requirements of CAAs and specify their scope and requirements. In addition, CAAs' functionalities need to be illustrated using new notations to express CIs and CSs. The notion of the context aware use-case diagram provides two concepts: the adjustable use case diagram and the use context diagram. However, the notations for both concepts are merged for this notion to express all functional requirements and context awareness requirements of CAAs and link them through a new <<utilize >> relationship.

This research has also extended the existing notations of the activity diagram to model the behaviour of CAAs and demonstrate their activities and adaptation actions. In addition, CAAs' behaviour needs to be demonstrated using new notations to express the activities of CAAs and CSs and their outputs through adjustable levels to depict the dynamic behaviour of CAAs in an Activity Diagram with new notations within CAA and CS swimlanes and their meta-swimlanes as controllers of CAAs' activities. A context aware activity diagram provides two levels of modelling: the first level is called high-level interactions (general modelling of CAAs' activities and swimlanes), while the second level is called low-level interactions (descriptive modelling of swimlanes'

CHAPTER 8. CONCLUSION AND FUTURE WORK

interactions and their meta-swimlanes). The context aware activity diagram also provides two concepts: the adjustable activity diagram and the context activity diagram. However, the two concepts' notations are merged for this notion to demonstrate all dynamic behaviour of CAAs' modelling.

Finally in this research, the existing elements of the class diagram have been extended to model structural components of CAAs and describe their context objects, classes and entities. In addition, CAAs' structure needs to be designed using new elements to express the structure of objects and contexts. The context aware class diagram notion provides two concepts: the adjustable class diagram and the context class diagram. However, the two concepts' elements are merged for this notion to design all component structures of CAAs and link them through a new relationship of <<utilization>>.

This thesis's contribution to knowledge can be summarised as follows:

- This research provides a thorough investigation of RE, UML and CAAs and also specifies the strengths and weaknesses of UML and its diagrams.
- It shows the interactions in clear models between stakeholders, CAAs and CI providers.
- It defines the functional requirements and the context awareness requirements of CAAs and their requirements using the notion of the context-aware use case diagram.
- It provides the behaviour activities of CAAs and their adaptation actions using the notion of the context-aware activity diagram.
- It specifies the object structure and context structure for CAAs using the notion of the context-aware class diagram.
- Finally, this research investigates the requirements for two real-world case studies of navigation system and weather forecast system using CA-UML diagrams.

8.5 Future Work

Today, intelligent systems and CAAs are helping human beings in different areas such as social life, travel, health, education and transportation. The services of CAAs face many challenges and issues in a wide range of subjects and still need further research and investigation. Modelling and design are important stages to define the real requirements of CAAs and users' needs.

My future work focuses on three areas of extending other UML diagrams, agile development, CCA, XML and complex smart systems such as SHS as follows:

- Extending other UML diagrams to model context-aware applications

UML diagrams include thirteen types of static and dynamic modelling. This research extended three types of use case, activity and class diagrams. My future work involves extending other UML diagrams, of which the most interesting types are sequence, package and object diagrams.

In addition, the main aim in my future plan is to extend other traditional UML diagrams to make comprehensive CA-UML diagrams for context-aware applications. These diagrams will provide new notations to represent the context-awareness requirements to be used for smart applications and systems. This extension of UML will resolve the limitations of UML and establish a clear usage that involves basic UML for traditional systems and CA-UML for intelligent systems. In other words, this future work will study other UML diagrams to be extended in an effort to facilitate the development of CAAs [87].

- Agile development for context-aware applications

Agile development is becoming an important methodology for different kinds of information systems. CA-UML diagrams have the ability to gather and specify the functional requirements and context-awareness requirements to be used in Agile development. In addition, Agile methodology can be evaluated by using the CA-UML approach to express standard processes, such as creating Agile models using a context-aware use case diagram for CAAs' functionalities.

CHAPTER 8. CONCLUSION AND FUTURE WORK

Agile development is suitable for responding to change: CAAs requirements are continuously changing and a new approach is necessary to specify the context-awareness requirements. These changes can be visualized by CA-UML diagrams to make clear Agile models for developers and providers of CAAs.

- Translating CA-UML diagrams into a CCA processes

CCA are an excellent process to translate CA-UML diagrams. This process can then be modelled and analysed using the CCA tools, such as the simulator ccaPL, which enables the execution of CCA processes, and the model-checker ccaSPIN. Model-checking tools such as ccaSPIN can also be used to analyse the functional requirements and context-awareness requirements of CAAs. My future work will focus on standard processes to translate CAAs' functionalities, behaviour and structure using context-aware use case diagrams, context-aware activity diagrams and context-aware class diagrams into a calculus of context-aware ambients.

- Extending XML to document context-aware web-applications

Today CAWA are widely used around the world, enabling users to be aware of different environmental objects. However, while UML and XML have become more usable for a set of application types and are suitable for intelligent systems as well as for XML documentation, they are still limited in terms of tagging all CAWA diagrams and models. The main objective of this investigation is to map the conversion between UML class diagrams and XML documents to specify the structure requirements of CAWA [52, 82].

The CAA development life-cycle is still lacking and requires investigation to be more efficient. XML is a documentation language which provides a clear definition of the structure and semantics of web applications as XML documents. XML was approved by the World Wide Web Consortium (W3C) as a W3C recommendation in May 2001. XML is widely used and accepted to represent information and as an internet sharing language. UML and XML technologies have grown up and been extended for traditional systems and are still able to model and document CAA functionalities.

CHAPTER 8. CONCLUSION AND FUTURE WORK

My future work involves extending XML to document CA-UML diagrams: this will help to convert the models from the design stage (context-aware class diagram notion) to the document stage in readiness for coding and implementation. The main objectives of this research are as follows:

- Suggest an extension approach of XML to illustrate the contextual information and its context providers affecting CAAs' services;
 - Propose a documentation approach for context-aware web applications, called CA-XML;
 - Map the CAAs' functionalities models from CA-UML diagrams to CA-XML documents;
 - Convert context-aware class diagrams to CA-XML documents as semantic webs;
 - Extend XML tags to document the parameters of inputs for CAAs' functionalities.
- Case Study of Smart Home System

The Smart Home System is an interesting case study to specify the exact requirements for collecting information for communication between a set of things in the home, as well as to monitor, warn and alert the user to changes of home components through mobile phone, internet and PC. In fact, the context-awareness requirements for a smart home system may be optional requirements for normal people but necessary for disabled people, especially those who are living alone. SHS gives more flexibility by integrating different units to react as one, which can be controlled automatically. Simulating the dynamic behaviour and designing the static structure of SHS is an important stage to define the specific users' needs and specify context-awareness requirements. However, the main task of SHS is environmental monitoring inside and around the house, while its biggest challenge is the behaviour awareness of all context changes in the home environment. Another task of SHS is to make life comfortable and facilitate control of various home units using a set of different types of CIs such as room temperature, humidity, lighting, alarms and kitchen gas using a set of different sensors to guarantee the family's happiness and security. SHS can also act as a burglar alarm system. In addition, SHS uses an

CHAPTER 8. CONCLUSION AND FUTURE WORK

internet connection to link the users with their home's equipment from anywhere in the world at any time [100, 102].

My future work will also investigate SHS requirements and the use of extension approaches of UML and XML for visualization, modelling, designing and documentation.

Bibliography

- [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 (Volume 353, pp 202-209), Korea, 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. Henriksen 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". ACM International Conference Proceeding Series; Vol. 183, 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, Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing, Georgia institute of technology, Atlanta, 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) 23rd International Conference on Software Engineering. IEEE Computer Society Press, London, UK, 2001.

BIBLIOGRAPHY

- [9] 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.
- [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] (UML) Unified Modeling Language diagrams, available at [<http://www.uml-diagrams.org/>].
- [18] M. Weiser. "The computer for the 21st century". SIGMOBILE Mob. Computer Community Rev. 3(3), 1999, pp. 3-11.

BIBLIOGRAPHY

- [19] Renaissance. "Technology briefing report system modelling". Technical report, School of Computing and Communication, Lancaster University, 1996, UK.
- [20] Hong, Chiu, Shen. "Requirements elicitation for the design of context-aware applications in a ubiquitous environment". Proceedings of the 7th international conference on Electronic commerce, 2005.
- [21] 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.
- [22] 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.
- [23] K. Alghathbar, C. Farkas and D. Wijesekera. "Securing UML information flow using flowUML". Research and Practice in Information Technology. 2006.
- [24] G. Sindre and A. L. Opdahl. "Eliciting security requirements with misuse cases". Requirements Engineering (2005) 10:34–44, DOI 10.1007/s00766-004-0194-4, London.
- [25] G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper and M. Pinkerton. "A mobile context-aware tour guide". Wireless Networks, 1997.
- [26] 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.
- [27] J. Jimenez and L. Iribarne. "Describing use-case relationships with sequence diagrams". The Computer Journal, 2007.
- [28] G. Anne Ngu. "Context aware actors". In Presented at the Ninth Biennial Ptolemy Miniconference, Berkeley, 2011.
- [29] 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.

BIBLIOGRAPHY

- [30] 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.
- [31] K. Peralta, P. Orozco, P. Zorzo, A. Oliveira. "Specifying security aspects in UML models". In 1st International Modeling Security Workshop. France, 2008.
- [32] T. Xiaosheng, S. Qinghua and Z. Ping. "A Distributed Context-Aware Model for Pervasive Service Environment". IEEE Computer Society, 2006.
- [33] M. Weiser. "Some Computer Science Issues in Ubiquitous Computing". ACM, 36:75-84, 1993.
- [34] 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.
- [35] P. Korpipaa, J. Mantyjarvi, J. Kela, H. Keranen and E. Malm. "Managing Context Information in Mobile Devices". Pervasive Computing IEEE, 2003.
- [36] Godbole, Smari. "Human perspective based context acquisition, learning and awareness in the design of context aware systems". Department of Electrical and Computer Engineering, University of Dayton, Military Communications Conference, 2006.IEEE.
- [37] B. Schilit, N. Adams, R. Want. "Context-Aware Computing Applications". IEEE Workshop on Mobile Computing Systems and Applications, 1994.
- [38] Kolos, Mazuryk, Poulisse, van Eck. "Requirements Engineering for Pervasive Services". 16 Octobre 2005. 2nd Workshop on Building software for pervasive computing.
- [39] J. Krogstie. "Requirement Engineering for Mobile Information Systems". Proceedings of the 7th International Workshop on Requirements Engineering, Interlaken, Switzerland, 2001.

BIBLIOGRAPHY

- [40] P. Loucopoulos, V. Karakostas. "System Requirements Engineering". McGraw-Hill International series in Software Engineering, 1995.
- [41] G. Chen, D. Kotz. "A survey of context-aware mobile computing research". Technical Report TR2000-381, Department of Computer Science, Dartmouth College, 2000.
- [42] T. Rodden, K. Chervest, N. Davies. "Exploiting Context in HCI Design for Mobile Systems". First Workshop on Human Computer Interaction with Mobile Devices, Glasgow, 21st & 22nd May 1998.
- [43] A. Dix, T. Rodden, N. Davies, J. Trevor, A. Friday, and K. Palfreyman. "Exploiting space and location as a design framework for interactive mobile systems". ACM Transactions on Human Computer Interaction, September 2000.
- [44] C. 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.
- [45] M. H. Al-Sammarraie. "Policy-based Approach for Context-Aware Systems". PhD thesis, De Montfort University, 2011.
- [46] S. M. Almutairi. "Modeling Security Requirements for Context-Aware Systems using UML". PhD thesis, De Montfort University, 2013.
- [47] H. M. Alotaibi. "Context-Aware and Secure Workflow Systems". PhD thesis, De Montfort University, 2012.
- [48] K. Alghathbar. "Enhancement of use case diagram to capture authorization requirements". Software Engineering Advances, International Conference on, 0:394-400, 2009.
- [49] K. Alghathbar and D. Wijesekera. "Modeling dynamic role-based access constraints using UML". In Proceedings of the 1st International Conference on Software Engineering Research & Applications (ICSERA 03), San Francisco, 2003.

BIBLIOGRAPHY

- [50] B. Farbey and A. Finkelstein. "Software Acquisition: a business strategy analysis". presented at Proceedings of Requirements Engineering (RE 2001), 2001.
- [51] K. Bittner and I. Spence. "Use Case Modeling". The Addison-Wesley Object Technology Series. Addison Wesley Professional, 2003.
- [52] C. Mascolo, W. Emmerich and A. Finkelstein. "XML Technologies and Software Engineering". Presented at Proceedings of the 23rd International Conference on Software Engineering, Toronto, Canada, 2001.
- [53] A. Finkelstein. "Software Engineering (Encyclopedia Entry) ". In Encyclopedia of Information Sciences: Marcel Dekker, 2000.
- [54] A. Finkelstein and J. Kramer. "Software Engineering: a roadmap". In The Future of Software Engineering, A. Finkelstein, Ed.: ACM Press, pp. 3-24, 2000.
- [55] B. Columbia. "Activity diagram modeling standards and guidelines version 1.0". Information and Technology Management Branch, December , 2005.
- [56] K. Breitman, J. Leite and A. Finkelstein. "The World a Stage: A Survey of Requirements Engineering Using a Real-life Case Study". Journal of the Brazilian Computer Society, vol. 6, pp. 13--37, 1999.
- [57] H. Alotaibi and H. Zedan. "The design and analysis of context-aware, secure workflow systems". In Proceedings of the 40th Informatics and Mathematics, 2010.
- [58] A. Al-alshuhai and F. Siewe. "An Extension of the Use Case Diagram to Model Context-aware Applications". Intelligent Systems Conference (SAI), London, UK, 10-11 November 2015.
- [59] A. Al-alshuhai and F. Siewe. "An Extension of the UML Activity diagram to model the behaviour of Context Aware Systems". The 15th IEEE International Conference on Computer and Information Technology (CIT-2015), Liverpool, UK, 26-28 October 2015.

BIBLIOGRAPHY

- [60] G. Booch, J. Rumbaugh and I. Jacobson. "Unified Modeling Language User Guide". The 2nd Edition (Addison-Wesley Object Technology Series). Addison-Wesley Professional, 2005.
- [61] G. Booch, J. Rumbaugh and I. Jacobson. "The Unified Modeling Language User Guide". 1999.
- [62] F. Siewe. "A Compositional Framework for the Development of Secure Access Control". PhD thesis, De Montfort University, 2005.
- [63] D. Salber, A. Dey and G. Abowd. "Ubiquitous Computing: Defining an HCI Research: Agenda for an Emerging Interaction Paradigm". Technical report, 1998.
- [64] A. Bouzeghoub, K. Do and L. Wives. "Situation-aware adaptive recommendation to assist mobile users in a campus environments". In International Conference on Advanced Information Networking and Applications, IEEE Computer Society, pages 503-508, 2009.
- [65] V. da Silva, R. Noya and C. de Lucena. "Using the UML 2.0 activity diagram to model agent plans and actions". In Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, AAMAS 05, pages 594-600, New York, USA, 2005.
- [66] G. Chafle, K. Dasgupta, A. Kumar, S. Mittal and B. Srivastava. "Adaptation in web service composition and execution". In International Conference on Web Services-ICWS, pages 549-557, 2006.
- [67] S. Poland. "Ubiquitous Computing: Smart Devices, Environment and Interactions". Wiley, 2009.
- [68] W. Emmerich. "Engineering Distributed Objects". JohnWiley and Sons, 2000.
- [69] M. Jackson. "The World and the Machine". In Proceedings of the 17th International Conference on Software Engineering, pages 283-292, Seattle, Washington, USA, 1995.

BIBLIOGRAPHY

- [70] W. Dargie. "Context-Aware Computing and Self-Managing Systems". Chapman & Hall/CRC Studies in Informatics Series. Taylor & Francis, 2010.
- [71] R. Milner. "Pure Bigraphs: Structure and Dynamics". Information and Computation/information and Control. 204:60-122, 2006.
- [72] R. Milner. "Bigraphs as a Model for Mobile Interaction". In Graph Transformation, volume 2505, pages 8-13. Springer Berlin/Heidelberg, 2002.
- [73] R. Milner. "Bigraphical Reactive Systems: Basic Theory". Technical Report 523, University of Cambridge, Computer Laboratory, 2001.
- [74] G. Chen, T. Finin and A. Joshi. "An ontology for context-aware pervasive computing environments". Knowl. Eng. Rev., 18 (3):197-207, 2003.
- [75] H. Hegering. "Management challenges of context-aware services in ubiquitous environments". In Proceedings of the 14th IFIP/IEEE Workshop on Distributed Systems: Operations and Management (DSCOM 2003, pages 246-259, 2003.
- [76] A. Day. "Understanding and using contexts". Personal and Ubiquitous Computing, 5 (1): 4-7, 2001.
- [77] S. Loke. "Context-Aware Pervasive Systems". Auerbach Publications, 2006.
- [78] S. Fickas and M. Feather. "Requirements Monitoring in Dynamic Environments". In Proceedings of the Second IEEE International Symposium on Requirements Engineering, pages 140–147. IEEE Computer Society Press, 1995.
- [79] K. Baina, B. Benatallah, F. Casati, and F. Toumani. "Model-Driven Web Service Development". In Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAiSE 04), Latvia, 2004.
- [80] H. Zedan, F. Siewe and A. Cau. "The calculus of context-aware ambients". Journal of Computer and System Science, 2010.
- [81] H. Zedan, F. Siewe and A. Cau. "The Validation of Context-Aware Systems". Technical Report STRL 08-47, Software Technology Research Laboratory, 2008.

BIBLIOGRAPHY

- [82] T. Strang and C. Linnhoff-Popien. "A Context Modeling Survey". In Workshop on Advanced Context Modelling, Reasoning, UbiComp 2004-The Sixth International Conference on Ubiquitous Computing, Nottingham, UK, 2004.
- [83] J. Subercaze, P. Maret, N. Dang and K. Sasaki. "Context-aware Applications Using Personal Sensors". In Proceedings of the ICST 2nd international conference on Body area networks, pages 19:1-19:5. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007.
- [84] S. Mellor, A. Clark, and T. Futagami. "Special Issue on Model-Driven Development". IEEE Software, 20(5):14–18, 2003.
- [85] C. Ptolemaeus. "System Design, Modeling, and Simulation". Using Ptolemy II, First Edition, Version 1.02, Ptolemy.org, ISBN: 978-1-304-42106-7, 2014.
- [86] G. Kortuem, Z. Segall and M. Bauer. "Context-Aware, Adaptive Wearable Computers as Remote Interfaces to Intelligent Environments". In Proceedings of the 2nd IEEE International Symposium on Wearable Computers, pages 58-65. IEEE Computer Society, 1998.
- [87] M. Keidl and A. Kemper. "Towards Context-Aware Adaptable Web Services". In Proceedings of the 13th International World Wide Web Conference (WWW 04), New York, USA, 2004.
- [88] L. Chung, B. Nixon, E. Yu and J. Mylopoulos. "Non-Functional Requirements in Software Engineering". Kluwer Academic Publishers, 2000.
- [89] M. Rosemann, J. Recker, C. Flender and P. Ansell. "Understanding context-awareness in business process design". In Proceedings of the 17th Australian Conference on Information Systems, 2006.
- [90] K. Khedo. "Context-Aware Systems for Mobile and Ubiquitous Networks". In Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies. IEEE Computer Society, 2006.

BIBLIOGRAPHY

- [91] P. Samarati, E. Bertino, A. Ciampichetti and S. Jajodia. "Information flow control in object-oriented systems". *IEEE Transactions on Knowledge and Data Engineering*. 9(4):524–538, 1997.
- [92] S. Clarke. "Composition of object-oriented software design models". PhD thesis, Dublin City University, 2001.
- [93] G. Biegel and V. Cahill. "A Framework for Developing Mobile, Context-aware Applications". In *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom 04), PERCOM 04*, pages 361-365. IEEE Computer Society, 2004.
- [94] T. Buchholz, A. Kupper, and M. Schiffers. "Quality of Context: What It Is And Why We Need It". In *Proceedings of the 10th Workshop of the OpenView University Association (OVUA 03)*, Geneva, 2003.
- [95] T. Doan, S. Demurjian, R. Ammar, and T. Ting. "UML design with security integration as a first class citizen". In *Proceedings of 3rd Intl. Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications (CSITeA 04)*, Cairo, 2004.
- [96] M. Beigl, A. Krohn, T. Zimmer and C. Decker. "Typical Sensors Needed in Ubiquitous and Pervasive Computing". In *Proceedings of the First International Workshop on Networked Sensing Systems (INSS 04)*, pages 153-158, 2004.
- [97] T. Doan, S. Demurjian, T. Ting, and A. Ketterl. "MAC and UML for secure software design". In *Proceedings of 2nd ACM Workshop. on Formal Methods in Security Engineering*, Washington D.C., 2004.
- [98] T. Lodderstedt, D. Basin, and J. Doser. "SecureUML: A UML-based modeling language for model-driven security". In *Proceedings of the 5th International Conference on The Unified Modeling Language*, pages 426–441. Springer-Verlag, 2002.
- [99] G. Roman, C. Julien and J. Payton. "Modeling adaptive behaviors in context unity". *Theoretical Computer Science*, 376(3):185-204, 2007.

BIBLIOGRAPHY

- [100] D. Harel. "Statecharts: A visual formulation for complex systems". *Science of Computer Programming*, 8(3):231-274, 1987.
- [101] N. Xu. "A Survey of Sensor Network Applications". *IEEE Communications Magazine*, 40(8), 2002.
- [102] J. Krishnamurthy, N. Narendra, K. Ponnalagu and R. Ramkumar. "Run-time adaptation of non-functional properties of composite web services via aspect-oriented programming". In *International Conference on Service-Oriented Computing (IC-SOC)*, 2007.
- [103] T. Arampatzis, J. Lygeros, and S. Manesis. "A Survey of Applications of Wireless Sensors and Wireless Sensor Networks". *IEEE International Sym. on Intelligent Control*, 2005.
- [104] R. Milner. "Communication and Concurrency". Prentice Hall, 1989.
- [105] R. Milner. "Functions as processes". *Journal of Mathematical Structure in Computer Science*, 2(2):119-141, 1992.
- [106] L. Fuentes and A. Vallecillo. "An Introduction to UML Profiles". *The European journal for the Informatics Professional*, 5(2), 2004.
- [107] Object Management Group. UML2.0 Super Structure Specification, 2004.
- [108] L. Ly, S. Rinderle and P. Dadam. "Integration and Verification of Semantic constraints in adaptive process management systems". Volume 64, pages 3-23, 2008.
- [109] H. Janicke, A. Cau, F. Siewe and H. Zedan. "A note on the formalisation of UCON". In *Proceedings of the 11th ACM Symposium on Access Control Models and Technologies (SACMAT07)*, pages 163-168, 2007.
- [110] H. Janicke, F. Siewe, K. Jones, A. Cau, and H. Zedan. "Analysis and Run-time Verification of Dynamic Security Policies". In Robert Ghanea-Hercock, Mark Greaves, Nick Jennings and Simon Thompson, editors, *Proceedings of the Workshop on Defence Applications of Multi-Agent Systems (DAMAS)*, pages 55-66, Utrecht University, The Netherlands, Springer, 2005.

BIBLIOGRAPHY

- [111] Kennedy Carter Ltd. The UML Action Specification Language Reference Guide, 2004.
- [112] E. Wilson. "An Introduction to Scientific Research". Dover books explaining science. Dover Publications, 1990.
- [113] G. Hynes, V. Reynolds and M. Hauswirth. "Enabling mobility between context-aware smart spaces". In International Conference on Advanced Information Networking and Applications, pages 255-260. IEEE Computer Society, 2009.
- [114] M. Heravizahed and D. Edmond. "Making workflows context-aware: A way to support knowledge-intensive tasks". In Proceedings of the 5th Asian-Pacific Conference on Conceptual Modeling, pages 231-240, 2008.
- [115] H. Baumeister, N. Koch, P. Kosiuczenko, and M. Wirsing. "Extending Activity Diagrams to Model Mobile Systems". In Proceedings of International Conference NetObjectDay (NODe) 02, LNCS 2591 Springer, Germany, pp. 278-293, 2002.
- [116] J. Harney and P. Doshi. "Speeding up adaptation of web service compositions using expiration times". In Proceedings of WWW 07, 2007.
- [117] P. Boonma and J. Suzuki. "An Adaptive, Scalable and Self-Healing Sensor Network Architecture for Autonomous Coastal Environmental Monitoring". IEEE Conference on Technologies for Homeland Security, 2007.
- [118] P. Stevens and R. Pooley. "Using UML software engineering with object and components". Addison-Wesley, 2006.
- [119] A. Al-alshuhai and F. Siewe. "An Extension of Class Diagram to Model the Structure of Context-Aware Systems". Sixth International Joint Conference on Advances in Engineering and Technology (AET-2015), NCR-Delhi region, India, 26 December 2015.
- [120] F. Siewe and A. Al-alshuhai. "From Use Case Diagrams to Executable Context-aware Ambients". The 10th International Conference for Internet Technology and Secured Transactions (ICITST-2015), London, UK, 14-16 December 2015.

BIBLIOGRAPHY

[121] F. Siewe, A. Al-alshuhai, S. Almutairi and A. Almutairi. "Analysing Use Case Diagrams in a Calculus of Context-aware Ambients". International Journal of Intelligent Computing Research, 2016.