Context-Aware and Adaptive Usage Control Model

Ph.D Thesis

Abdulgader Z. Almutairi

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

September 2013

Dedication

To my father

Zaid Almutairi

For all his prayers, sacrifices without his endless support

I could not have accomplished my Thesis

To my mother

Nawir Almutairi

For her endless love, encouragement and prayers

Thank you very much indeed for everything you have done for me

To my wife

Asma Almutairi

For her endless love, support, and patience

Without her patience most of these work would not have been done

Abstract

Information protection is a key issue for the acceptance and adoption of pervasive computing systems where various portable devices such as smart phones, Personal Digital Assistants (PDAs) and laptop computers are being used to share information and to access digital resources via wireless connection to the Internet. Because these are resources constrained devices and highly mobile, changes in the environmental context or device context can affect the security of the system a great deal. A proper security mechanism must be put in place which is able to cope with changing environmental and system context.

Usage CONtrol (UCON) model is the latest major enhancement of the traditional access control models which enables mutability of subject and object attributes, and continuity of control on usage of resources. In UCON, access permission decision is based on three factors: authorisations, obligations and conditions. While authorisations and obligations are requirements that must be fulfilled by the subject and the object, conditions are subject and object independent requirements that must be satisfied by the environment. As a consequence, access permission may be revoked (and the access stopped) as a result of changes in the environment regardless of whether the authorisations and obligations requirements are met. This constitutes a major shortcoming of the UCON model in pervasive computing systems which constantly strive to adapt to environmental changes so as to minimise disruptions to the user.

We propose a Context-Aware and Adaptive Usage Control (CA-UCON) model which extends the traditional UCON model to enable adaptation to environmental changes in the aim of preserving continuity of access. Indeed, when the authorisation and obligations requirements are fulfilled by the subject and object, and the conditions requirements fail due to changes in the environmental or the system context, our proposed model CA-UCON triggers specific actions in order to adapt to the new situation, so as to ensure continuity of usage.

We then propose an architecture of CA-UCON model, presenting its various components. In this model, we integrated the adaptation decision with usage decision architecture, the comprehensive definition of each components and reveals the functions performed by each components in the architecture are presented.

We also propose a novel computational model of our CA-UCON architecture. This model is formally specified as a finite state machine. It demonstrates how the access request of the subject is handled in CA-UCON model, including detail with regards to revoking of access and actions undertaken due to context changes. The extension of the original UCON architecture can be understood from this model.

The formal specification of the CA-UCON is presented utilising the Calculus of Context-aware Ambients (CCA). This mathematical notation is considered suitable for modelling mobile and context-aware systems and has been preferred over alternatives for the following reasons: (i) Mobility and Context awareness are primitive constructs in CCA; (ii) A system's properties can be formally analysed; (iii) Most importantly, CCA specifications are executable allowing early validation of system properties and accelerated development of prototypes.

For evaluation of CA-UCON model, a real-world case study of a ubiquitous learning (u-learning) system is selected. We propose a CA-UCON model for the u-learning system. This model is then formalised in CCA and the resultant specification is executed and analysed using an execution environment of CCA.

Finally, we investigate the enforcement approaches for CA-UCON model. We present the CA-UCON reference monitor architecture with its components. We then proceed to demonstrate three types of enforcement architectures of the CA-UCON model: centralised architecture, distributed architecture and hybrid architecture. These are discussed in detail, including the analysis of their merits and drawbacks.

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.
This thesis is written by me and produced using LaTeX.

Abdulgader Almutairi

Publications

1. A. Almutairi, and F. Siewe. CA-UCON: a context-aware usage control model. *Proceedings of the 5th ACM International Workshop on Context-Awareness for Self-Managing Systems*, CASEMANS'11 - Beijing, China, pages 38–43, 2011.

2. A. Almutairi, and F. Siewe. Formal Specification of CA-UCON model using *CCA. Science and Information Conference 2013.* 7-9 October 2013, London, UK.(Accepted).

3. A. Almutairi, and F. Siewe. Modelling Usage Control of a U-learning System using CA-UCON. *The 4th Workshop on Service Discovery and Composition in Ubiquitous and Pervasive Environments (SUPE'2013).* August 26-28, 2013, Cyprus.

4. A. Almutairi, and F. Siewe. Enforcement of CA-UCON Model. *The seventh International Conference on Mobile Computing and Ubiquitous Networking.* January 6-8, 2014, Singapore Management University, Singapore. (Submitted).

5. W. Alkhaldi. S. Almutairi. A. Almutairi and K. Aldrawiesh. Toward Development Context Aware Advertisement system (Case Study). *in proceedings of the IEEE 2011 International Conference on Computer Applications and Network Security (ICCANS 2011)*, May 27th-29th 2011 in Maldives, IEEE Press, CFP1182M-PRT/ISBN: 978-1-4244-9764-5.

Acknowledgments

First and foremost, I am grateful to almighty and the most merciful **ALLAH** for giving me the strength and courage and for enabling me complete this thesis, without whom nothing is possible.

My sincerest thankfulness and deepest appreciation goes to my mentor, great supervisor **Dr Francois Siewe** for his guidance, care, concern throughout my PhD study; without his support, encouragement and guidance this thesis would not have been possible to achieve. I am really happy that I was able to finish my PhD under his supervision.

I also, would like to express my deepest thanks to **Prof. Hussein Zedan**, the head of the STRL for his guidance, love, and care for everyone in STRL. I would like to thank **Dr. Ali Al-Bayatti** for his guidance, insightful suggestion in this work and motivation.

I would like to thank all researchers, colleagues and staff of the STRL for the friendly and convenient working environment I could experience there.

On family side, I would express my deepest thanks to my parents for their prayers, support and encouragement, whose prayers and blessings were no doubt the true reason behind any success I have.

Finally, I would like to special thank my wife (Asma) and my children (Abdulaziz, Rana and Ahmed) for being patient while I was doing my thesis, without their patience most of these work would not have been accomplished.

Table of Contents

List of Figures

List of Tables

List of Abbreviation

UCON	Usage control
CA-UCON	Context-aware usage control
CCA	Calculus of Context-aware Ambients
U-learning	Ubiquitous Learning
RM	Reference Monitor
DAC	Discretionary Access Control
MAC	Mandatory Access Control
RBAC	Role-Based Access Control
TM	Trust Management
ccaPL	Calculus of Context-aware Ambients Programming Language
DRM	Digital Right Management
CE	Context Expression
Ubicomp	Ubiquitous computing
UD	Usage Decision
AD	Adaptation Decision

PreA	Pre-Authorizations
OnA	On-Authorization
PreB	Pre-oBligation
OnB	On-oBligation
PreC	Pre-Condition
OnC	On-Condition
PreD	Pre-Adaptation
OnD	On-Adaptation
GC	Garbage collector
HB	High Bandwidth
FMem	Free Memory
AC	Access Control
ICT	Information Communication Technology
A	Authorizations
B	oBligation
C	Condition
D	Adaptation
R	Set of access Rights
S	Set of Subjects
O	Set of Objects

Att(s)	Attributes of the subject s
Att(o)	Attributes of the object o
PCS	Personal Communications Services
UML	Unified Modelling Language
ER	Entity Relationships
CoBrA	Context Broker Architecture
CAAC	Context-aware Access Control
GRBAC	Generalised-Role-Based Access control
TRBAC	Temporal Role-Based Access control
SRBAC	Spatial Role-Based Access Control
GTRBAC	Generalised Temporal Role-Based Access Control
DRBAC	Dynamic Role-Based Access Control
CA	Central Authority
TUCON	Times-Based Usage Control
GEO-UCON	Geography Usage Control
ConUCON	Context-aware Usage control
CUC	The Contextual Usage Control
FMS	Finite State Machine

Chapter 1

Introduction

Objectives:

- Present an introduction and motivation of this research.
- List the research questions.
- Present the research methodology.
- Give the thesis structure.

1.1 Introduction and Motivation

Ubiquitous environments use a wide variety of devices such as mobile phones, laptops, wearable computers, embedded computing devices and hardware sensors. These devices employ wireless and/or wired networks, which have differing limitations in terms of power consumption, memory size and computational speed [2]. These networks can be incorporated and interact with each other in order to produce adaptive services, which are provided depending on the present context of the user, the environment and the system. Many new applications are now taking advantage of the ubiquitous computing paradigm, facilitating the creation of smart homes, health systems, smart vehicles, etc. One of the most significant features of ubicomp is that it is 'context-aware' [57].

The concept of a context-aware system is that the system is able to sense its surrounding environment (the context) and can react accordingly. The widely accepted definition of context was given by [24] 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." Thus, the three functions that must be incorporated into a context-aware system are: (i) sense the environment, i.e. context; (ii) process any change in the context; and (iii) adapt to the new context.

Various modern technologies are being exploited in networks and computing, leading to the further development and increased use of ubicomp, and facilitating a variety of 'convenience applications' [77]. However, this increase in ubicomp has introduced new security challenges, as information can be accessed and shared by

users anytime and anywhere. Therefore, access through adaptive services and smart devices must be protected by an efficient access control system. Traditional access control models, such as Discretionary Access Control (DAC), Mandatory Access Control (MAC) and Role Based Access Control (RBAC), were designed for traditional computing environments and accordingly have certain boundaries; these limit their adoption and utilization in ubicomp environments. For example, the access decision in traditional access control models is based on static attributes, which makes them unsuitable for dynamic environments where the context of the user and environment must be considered in any access decision [102].

The Usage Control (UCON) model is the latest major enhancement to traditional access control models; it enables mutability of subject and object attributes, and continuity of control on usage of resources [73]. UCON has a condition component, which can sense and capture the context, but it cannot react and adapt to the new situation depending only on this context. Therefore, UCON is not wholly appropriate for dynamic environments such as in ubicomp, where the context can change frequently; this would affect the access decision for a subject who requests access to an object. Thus, UCON needs to be further enhanced to make it an adaptive model, i.e. one that interacts with the changing environment and adapts to any new situation in order to ensure continuity of service.

In this motivated example, We assume a vehicle equipped with smart system that is able to connect wirelessly to any network and permits user to download music or video services. We assume that the system is able to sense the bandwidth of network connection when the user requesting the downloading music or video services .So the condition for downloading this service is that the bandwidth must be high. If the user wants to download the service, the system will check the authorization

and the obligation and the condition of the current request. So if the authorization and obligation are met but the condition is not met which means in this case the bandwidth is low, the system will deny the access. In this case the need for adaptive model that adapts to new situation by do the action (e.g. slowdown the car speed) in order to get the high bandwidth and continue the access to the user is important.

1.2 Problem Description and Research Question

Producing an adaptive usage control model that is compatible with UbiComp environments is a challenging task. A ubicomp system must be able to adapt its behaviour based on its environment. Thus, it must respond to change in current context if it is to deliver an adaptive service. The security system that is utilized in an ubicomp environment (in order to control access in adaptive services) must accordingly be context-aware.

Many access control models have been proposed to suit different aspects of ubicomp environments. Most of the proposed models and architectures have been built on traditional access control models, such as RBAC [26][62]. Some researches have been conducted in an attempt to extend the UCON model in order to build an adaptive model that is appropriate for the ubiquitous environment [113][39].

However, to the best of our knowledge, none of the existing extensions to access control models, particularly UCON, has completely solved this issue in the ubiquitous environment, where context-awareness is a vital aspect of any ubicomp system, as the context is constantly changing. Thus, it is essential that an adaptive access control model (that is better suited to the ubiquitous environment) be developed in

order to remedy the limitations of traditional access control models.

In this thesis, we present a new usage control model, one that is adaptive and context-aware, called *Context-Aware and Adaptive Usage Control* model (CA-UCON in short). This new model ensures three features: (i) data protection, (ii) enhanced service quality; e.g. ensuring the continuity of usage despite change in the environment and (iii) keeping explicit interactions with the user at a minimum.

The overall and key research question can thus be presented as follows:

How can an adaptive usage control model (context-aware) solution, one that seamlessly suits a dynamic computing environment (ubicomp), be produced?

The main aim of this research is to address the question above in an efficient manner. However, in order to answer this research question, three sub-questions can be formulated as follows:

- How can the adaptation process be integrated into the usage control model?
- How can the new model be formally analysed using a suitable formal notation?
- How can an adaptive usage control model be enforced in ubiquitous environments?

1.3 Research Methodology

The research methodology employed in this thesis is a standard scientific research technique (Constructive method), in which novelty is developed through an innova-

tive architecture, model or technique. Having expert knowledge in this particular field is vital if novelty is to be developed, and the literature will be key to this. Accordingly, the proposed approach is comprised of five work packages; the first is a literature review, and the second focuses on the proposed architecture. The third work package presents the suitable formal specification, and the fourth demonstrates different enforcement architectures for an adaptive model. The last work package concentrates on the evaluation of the work [42].

- **Work package 1:** Research background

  The research background mostly comprises a literature review, which involves assessing all the aspects associated with the research question. Various resources are utilized, such as digital libraries, publications, journals, articles, etc.

- **Work package 2:** Architecture

  This work package presents the proposed architecture to capture the aim of this research, as formulated in the research question. Throughout this phase, the research explores context-aware access control models and their boundaries in order to identify the objectives and requirements of the proposed architecture; it also describes its contribution to the field.

- **Work package 3:** Formal specification

  This work package explores the available formal methods in order to generate specifications for the context-aware and adaptive usage control model, and elaborates the appropriate mathematical notations. Thus, the selected formal method is utilized to specify various aspects associated with the remit of this thesis.

- **Work package 4:** Enforcement architecture

In this phase, the research investigates the different enforcement architectures on which the adaptive model can be enforced. Then, it studies the most appropriate reference monitor and explains its components.

- **Work package 5:** Evaluation
  In the final phase, a real-world case study of ubiquitous learning is selected, and the discussed formal method is applied to formalise this case study. The properties of the context-aware and adaptive usage control model are analysed, through the execution environment of the aforementioned formal method.

1.4 Criteria for Success

Satisfying the criteria of the research in this thesis is achieved as follows:

- The research questions posited at the commencement of this work must be met.
- A study demonstrating how our proposed architecture is distinct from others must be presented.
- An analysis of the proposed system using *CCA*.
- A study that illustrates the appropriate enforcement architecture must be presented.

1.5 Thesis Structure

This section describes the outline of the remaining chapters of this thesis based on the aims of this research.

Chapter 2: Literature Review

Firstly, an overview of traditional access control models is presented in this chapter. It begins with a definition of access control, describing different access control models, and explains their differences. This chapter then illustrates the Usage Control (UCON)model and investigates the distinction between this model and the traditional access control models. Moreover, it examines the conceptual model and the family-core model of usage control. Subsequently, it provides an overview of ubicomp systems and context-aware systems. It provides a definition for context, and examines context models, context-aware systems and the properties of ubiquitous systems. In addition, a definition of adaptive systems and adaptation in ubicomp systems are presented. Then a various aspects of adaptation approaches are presented. Finally, related works on context-aware access control are presented. Different attempts to extend access control models, particularly UCON model are assessed.

Chapter 3: Context-Aware and Adaptive Usage Control Model (CA-UCON)

This chapter proposes a novel context-aware and adaptive usage control architecture, which extends the Usage Control (UCON) model. It enables the adaptation to environmental changes with the aim of preserving continuity of usage in a pervasive computing system. It then describes the computational model of CA-UCON as a Finite State Machine (FMS), depicting how a subject's request to access an object is handled in this model. This chapter also provides formal definitions of the two newly introduced adaptation models within the CA-UCON model. Finally, it presents the expressive power of CA-UCON, which the UCON model can be specified in CA-UCON model, and so all the access control models that can be represented

in UCON (such as RBAC, MAC, DAC and DRM)

Chapter 4: Formal Specification of the CA-UCON Model

This chapter is separated into three parts. The first part illustrates the selected formal method; it presents *CCA*. *CCA* has been proposed and is considered an appropriate mathematical notation for modelling mobile applications that are context-aware. The second part shows the ambient-based model within CA-UCON, which can model any entity in a system as ambient. The final part presents the formalisation of the CA-UCON model in *CCA*.

Chapter 5: Case Study

This chapter is based on a ubiquitous learning (u-learning) system as a case study; it explains in detail the requirements of a u-learning system. These requirements are then modelled in CA-UCON. The u-learning system is then formalised in *CCA* and executed. It is used to analyse the behaviour of the underlying CA-UCON model. These are done via devising scenarios in a u-learning system and executing them in order to assess the behaviour of CA-UCON in a changing context.

Chapter 6: Enforcement Architecture

This chapter presents the architecture of the CA-UCON Reference Monitor (CA-UCON RM) and explains all included components. Then, it investigates the enforcement architecture of the CA-UCON model by demonstrating three types of enforcement, which are known as Centralised Enforcement Architecture, Distributed Enforcement Architecture and Hybrid Enforcement Architecture. Moreover, this chapter details the differences between these architectures and presents the advantages and disadvantages of each architecture. Finally, some examples are illustrated in order to present the efficacy of the enforcement within CA-UCON under different

architectures.

Chapter 7: Conclusions and Future Work

This chapter presents a summary of the research in this thesis as well as the final results. Then, it suggests future work that may build upon the boundaries of our model.

Chapter 2

Literature Review

Objectives:

- Present a literature on Traditional Access Control Model.
- Investigate Usage Contol Model (UCON).
- Explore Pervasive (Ubiquitous) Computing, Context,Context-aware System.
- Show Adaptive system, Adaptation in Ubiquitous system, Adaptation approaches.
- Present some existing Context-aware access control model.

2.1 Introduction

In this chapter we give an account of background of the area of access control models. We begin with a definition of access control, introducing a traditional access control models, and follow with an explanation of the distinction between these models. In addition, we illustrate the usage control model (UCON) and describe the differences between this model and the traditional access control models. The conceptual model and family-core model of UCON are presented. An overview of Ubicomp systems and context-aware systems are shown, including the definition of context, context model and context-aware systems. Then, we present a different adaptation approaches and make the comparison between these approaches in order to select the most suitable approach for our work. Finally, related work on extension of access control to include context information in access decision are presented.

2.2 Background

Important new challenges have emerged as a consequence of recent technological innovations and advancements in the telecommunications, computing and networks. Various devices, such as smart phones, personal computers and portable devices are now being used to share resources and digital information, which make it difficult to protect such shared digital resources against unauthorized accesses [12].

Researchers in the relevant areas have attempted to mitigate and resolve the unauthorized access problem, and have made significant advances in the protection of scientifically oriented data and information as well as in the overall information security domain. Since the advent of the information security discipline, the infor-

mation security community has considered 'access control' as a major information security issue. Traditionally, access control concentrates on safeguarding computational resources against unauthorized access in a closed environment. The control mechanism employed focuses primarily on identity and attributes of the recognized client, or uses a reference monitor with specified authorization rules [72].

There are many ways to define access control; however, we can define it as the ability to grant or deny access to a subject to a particular object or resource in order to ensure protection against unauthorized access to particular digital information or computer resources. Hence, the objective of access control is to ensure three elements [53]:

Confidentiality: prevention and mitigation of the unauthorized exposure of a digital resource.

Integrity: prevention and mitigation of unauthorized modification to digital resources that could undermine their integrity.

Availability: ensuring access for authorized and authenticated subjects .

There are three main traditional access control models:

1. Discretionary Access Control (DAC).
2. Mandatory Access Control (MAC).
3. Role Based Access Control (RBAC).

These models were developed based on a set of predefined authorization rules, comprising 'subject', 'object' and 'operation' for each and every entity in a digital resource that needs access protection. Basically, these define what resources (objects) a user (subject) can or cannot access, and what actions the user is authorized and permitted to execute (operation) on the resources that they are permitted to access [94]. The focus of the traditional access control models is to protect digital and computational resources within a closed security environment. However, the recent rapid advances in complex computing systems have necessitated the introduction of new security requirements, and hence the need for a more robust new security mechanism [85].

The traditional security models do not and cannot sufficiently address all the major problems and challenges that have arisen in modern computer network systems. More robust and flexible models are needed to ensure access protection for modern digital resources. Furthermore, DAC is mainly responsible for controlling access to objects; it does not control the information flow. MAC can control the information flow but may not be suitable for controlling the access to objects. RBAC can control and administer access to digital information and other computational resources in a closed and controlled organizational domain but may not be able to do so if subjects are unknown in an open environment systems [95].

Another model, Trust Management, can be used in open environment systems to control and authorize access for unknown subjects only when the entities accessing are static and do not often change. Trust Management and the other traditional access control models control access to objects on the server side [104]. Moreover, studies have been conducted on access control issues and the usage of digital objects before and after dissemination; this is known as Digital Rights Management (DRM).

DRM opens up many opportunities for the commercial digital-information sector. Hence, the majority of current DRM solutions concentrate on payment-based dissemination control but can be applied as well on non-payment based dissemination [12]. However, as DRM and Trust Management have each focused on their own specific issues in terms of access control, a new Usage Control (UCON) access model was recently proposed. UCON extends multiple aspects of the traditional access control models, introducing two new aspects: mutability of attributes, and continuity of the access permission decision [83].

One of the most rapidly developing areas in ICT is known as 'ubiquitous computing'. It refers to the ever-increasing phenomenon of integrating and embedding ICT tools in people's daily lives and in the situations or environment in which they live. This has been made possible by the ever-improving developments in the manufacture of microprocessors, which now have built-in communication functions and other amenities [77].
The new technologies that have being employed in networks and computing have led to the development of ubiquitous computing, facilitating a variety of convenient applications [77]. The most significant matter in ubiquitous computing, which brings new challenges, is security, where the information can be accessed and shared by users anytime and anywhere.

Many researchers have attempted to solve security issues in ubiquitous environments, but none of them completely solves these problems. Thus, a new and effective security mechanism must be put in place; one that is able to cope with changing environmental and system contexts.[102]

2.3 Traditional access control models

Throughout the history of computing and information security, there have been several studies and efforts to facilitate and protect the access and the usage of digital resources. As above and now in Figure 2.1, traditional access control models can be divided into three: Mandatory Access Control (MAC), Discretionary Access Control (DAC), and Role-Based Access Control (RBAC)[86]. Despite the success of these models in closed environments, the issue of access control remains a major issue for information and system security in our modern cyberspace. Providers of digital content services and other relevant resources need to exercise control over accessibility and usage of their resources in the sense of gaining the ability to determine who is allowed to access a digital resource and what access rights they can have over the objects available; this is the central issue in any access control model. [95].

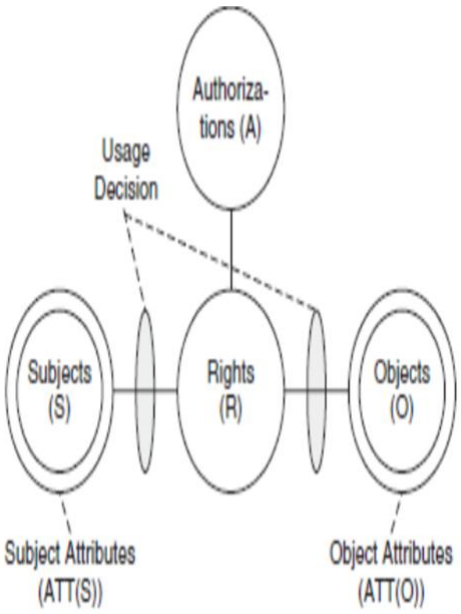


Figure 2.1: Traditional Access Control [86]

2.3.1 Discretionary Access Control (DAC)

Discretionary access control is defined as a denotation of restriction access to the available resources or services, relying on the identification of authorized users [55]. Most operating systems today use the DAC model at the heart of their controlling mechanisms. DAC provides flexibility in access control by devolving the protection decision and the right to determine access to the user, who owns or has created the object. The flexibility of DAC enables the object owner to discretionarily specify the type of access privileges for a potential user. However, "any practical DAC scheme must have significant constraints which can be best be understood in terms of the layered construction of protection systems" [97]. In general, discretionary access control models are concerned primarily with controlling users' access to restricted information resources.

2.3.2 Mandatory Access Controls (MAC)

Mandatory access control is a process, acting to restrict a user's access to resources based on 'access sensitivity', which is defined for the object in terms of the information objects that those resources contain; this entails a more formal authorization in the form of clearance, which is needed by users to access such sensitive objects [103]. "MAC bases the access rights on fixed regulations determined by a central authority" [95]; it is concerned mostly with the control of information flow between the different objects in a system. Moreover, the MAC security model is deemed to have a better access control in terms of security to objects than DAC, as MAC can control "indirect information flow". [28].

2.3.3 Role Based Access Control (RBAC)

In RBAC, users are given predefined access rights in the form of roles, which they can assume and execute on the system; hence, this restricts a user's access to other objects, i.e. those that do not concern them or those for which access has not been assigned. As the concept of 'role' stems from individuals in an enterprise or organization, where each and every person has a specific role and responsibility, RBAC facilitates modelling object access security from an enterprise or organizational perspective, by aligning the assigned roles with the roles that a subject is assigned to within an organization or enterprise [68]. RBAC is used in commercial applications as a viable alternative to the traditional models of mandatory/discretionary access control, as it can align users' access rights in a given system based on their assigned responsibility in the organization [89][88].

2.4 Trust Management (TM)

Trust Management is another access control model, which is used in authorizing unknown entities to have access to objects in an open environment. However, Trust Management is most generally applied when the entities are static, which are not likely to change over time. The concept of Trust Management was first proposed by [13]. It is described as a process of "using a uniform method that describes and explains security policy, security credential and trust relationship which is used to directly mandate to the key safety operation" . Moreover, the widely accepted definition of trust management was proposed by [76], stating that it "refers to the acquisition, evaluation, and implementation of trust intention". Based on this definition, the trust of subject in the object might be assessed utilizing mathematical process rely on previous experience and can be amended continuously based on the

behavioral result of object.

2.5 Digital Right Management (DRM)

DRM is one of many technology-oriented access control models. With the deployment of DRM, content owners can restrict and protect their digital contents before distributing and forwarding them by creating usage rules for those contents[56]. Such usage rules may include assigning temporary versions (trial), content expiration, content subscription, etc. DRM can be highly effective in protecting shareable and distributable digital information in an open environment [52] [103].

2.6 Usage Control (UCON) Model

2.6.1 Introduction

This section discusses the basic principles of a new access control, known as Usage Control (UCON); it was originally proposed by [73] to address issues in, and to provide various access control systems for, the new and modern computer applications environment. UCON uses attributes to determine access permission to a controlled digital information or computational resource based on three factors: authorization, obligations and conditions. These elements shall be discussed in detail in the subsequent sections. Moreover, UCON provides enhancements to existing traditional access control models by taking into consideration two new aspects for any object requiring access: mutability or changeability of attributes, and continuity of the access permission decision. The concept of mutability refers to the fact that attributes are not static; rather, they change intermittently, and hence the access permission decision should be dynamic and continually re-evaluated and updated (as soon as a new circumstance arises). The continuity of access decision ensures that any de-

cision to permit and allow access to an object is made constantly both before and during the access to an object [72].

2.6.2 Usage Control Conceptual Model

The Usage Control (UCON) model is known as a conceptual framework that provides a unified structure to protect digital resources. The UCON framework is not introduced to substitute or replace existing traditional access controls, Trust Management or DRM. Instead, the UCON framework provides enhancements and thus includes the existing access controls by applying two new access control concepts mentioned above: mutability of attributes and continuity of access decision [78]. The continuity of access decision in UCON ensures that the security policy is enforced before and during access has been granted to an object. The UCON authorization mechanism ensures that the granted usage access is revoked and terminated if the object's attributes change while the access is in progress and if the security policy in place is not satisfied with any particular attribute change [112]. As mentioned above, three factors are used to determine the access decision in the UCON model: authorization, obligations and conditions. Authorization is a predicate that enforces constraints on the attributes of both the subject and object when the subject wants to access an object. Obligations are the expected sequence of actions that are required by the subject before and during the usage of a given object. Conditions are a set of restrictions that are dependent on the environment of the access, which are expected to be valid at the time of access and during the access period.

As an illustration of the three factors discussed earlier, a UCON security and access policy may enforce aspects such as ensuring that a subject's name is something specific (i.e. authorization), that the subject may be required to sign an agreement form before access, which must not be violated (i.e. obligations), and finally that the object that needs to be accessed may only be accessed during official working

hours (i.e. conditions) [110]. To design such an access control system, the first action required is to identify the objects that need to be protected, then to identify the subjects that could request usage or operation on the objects that have been identified, and to identify all the possible actions that can be operated and executed on those objects, and finally to identify all the access rights needed to execute the identified actions on the objects [111].

2.6.3 Usage Control Components

The UCON$_{ABC}$ model comprises eight key components (see Figure 2.2), namely: subjects, objects, subject attributes, object attributes, authorizations, rights, obligations and conditions. Usage decisions are based on authorizations, obligations and conditions; these are functional predicates that must be evaluated for each usage decision [72].

2.6.3.1 Subjects (S) and Subject Attributes (ATT(S))

A subject is an entity who requests access to a resource; it must hold certain rights of access to the target object or resource, must be able to commence the request for usage, and consequently must be capable of executing the rights granted and assigned to it . The subject or resource requester is mostly defined and represented by its attributes, i.e. subject attributes [37]. The subject attributes, ATT(S), are basically the capabilities and properties that describe the subject, which can be used as a basis for any possible usage decision [79].

2.6.3.2 Objects (O) and Object Attributes (ATT (O))

Objects are basically the entities of interest to which subjects have predefined rights of access and use. Object attributes, ATT (O), are descriptions and properties of a given object, which could be used as basis in the provision for and making usage

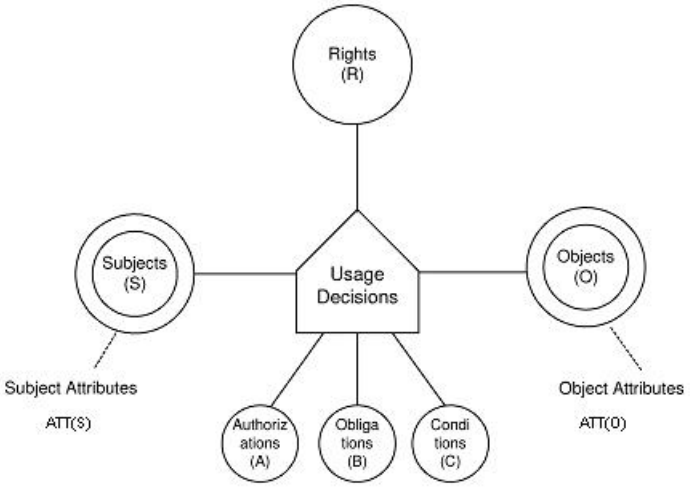


Figure 2.2: Usage Control Model [72]

decisions. The target objects to which subjects hold rights can be of a variety of types: an object may be digital information, or computational, network or service-oriented resources [40]. An object may be modified, shared or distributed by subjects in an open computer network environment. Objects may also be original, derivative or copied, or may be privacy or security sensitive (or otherwise). For example, access usage histories of subjects are considered as derivative of an object. In UCON, non-original objects (derivatives) are generated as result of using the original copy [38].

2.6.3.3 Attributes

Attribute is a commonly used concept in access control. For example, in the MAC model, a security tag assigned to an object can be considered as an object's attribute and a user role is also considered a subject's attribute in the RBAC model. The mutability of attributes is also introduced in the current access control approaches, but in most cases, a systems administrator control and force attribute updates. The mutability of objects and subject attributes resulting from UCON usage processes is considered the backbone of the UCON model. There are three different types of attributes in UCON, namely: environmental attributes, object attributes and sub-

ject attributes. Subject and object attributes were discussed above. Environmental attributes are considered as system-oriented attributes, such a device platform, area code, etc [74].

2.6.3.4 Rights

The concept of access right in UCON is the same as in traditional access control models. The access rights granted to subjects are not predefined but rather are granted on demand when a subject attempts to access an object, and the access right granted is based on the subject itself as well as the object, the authorization, the obligations, the conditions and the environmental attributes [60].

2.6.3.5 Authorizations (A)

Authorizations are defined as a key functional requirement that must be satisfied before granting a particular right of access to a digital object. There are various types of authorizations, and broadly they are utilized in many different contexts, including in the traditional access models. For instance, an authorization predicate may require that a user who wants to access an object or resource must be 22 years old or more, etc. Authorization predicates place conditions and constraints in the form of logical predicates on both the subject and object attributes. Unlike traditional models (which use pre-authorization), UCON authorization predicates are triggered and evaluated both before (pre-authorization) and during (on-authorization) the subject's execution of assigned rights [85].

2.6.3.6 oBligations (B)

Obligations are also defined as functional predicates; they are used to confirm the compulsory requirements that a subject must undertake before and during a particular usage process. The mandatory requirements here may be either pre-obligations

(preB) or ongoing-obligations (onB). preB uses a Boolean function and returns either 'true' or 'false' in order to determine whether or not certain pre-requisite actions have been fulfilled before granting access; for instance, a user agreeing to provide a log before listening to a music file, or a subject agreeing to provide personal information on a form before reading a company's free whitepaper, etc. onB represents the ongoing-obligations that a subject must fulfil in order to ensure that a particular predicate is active (continuously or periodically) while the permitted rights are being executed or are in use. For example, a subject may be required to keep a particular online advertisement active on the side while the subject is still logged on to watch a free video. Subject or object attributes may or may not be used by obligations. Attributes are useful for determining the types of obligations that may be required for usage approval. In some cases, obligations require making certain updates on subject attributes; those updates would likely affect both future and current usage decisions. However, attributes are not utilized in making decisions in relation to obligations; rather, they are only used to determine which obligations to select and perform [112].

2.6.3.7 Conditions(C)

Conditions are defined as environmental restrictions that must be considered in the process of a usage decision. They are not related directly to objects or subjects but they do rely on environmental properties or attributes. The assessment of condition predicates may take place before granting access permission to a digital object (pre-condition) or while the subject is using the object (on-condition). However, conditions do not update object, subject or environmental attributes, and the status value of conditions may be altered as a consequence of environmental amendment [78].

2.7 The UCON$_{ABC}$ family core modules

In this section, we briefly discuss UCON$_{ABC}$ family core models, which rely on the combination of three decision factors described above (authorization, obligations and conditions), as well as on continuity of usage decision and mutability of attributes. These core models were developed for the simple reason that they concentrate largely on the enforcement process of usage decisions, rather than on administrative issues. The ABC model functions (with the assumption of a usage request on a target object and the decision to grant access rights) can be performed either before or during the action and execution of requested rights. As a side effect of usage decision, mutability of objects permits limited updates on a subject or object attribute. Therefore, the UCON model identifies six core models, known as pre-authorization (UCON$_{PreA}$), on-authorization (UCON$_{OnA}$), pre-obligation (UCON$_{PreB}$), on-obligation (UCON$_{OnB}$), pre-condition (UCON$_{PreC}$) and on-condition (UCON$_{OnC}$). The models incorporate five components: subjects (S), objects (O), rights (R), subject attributes (ATT(S)), object attributes (ATT(O)). The notation *allowed(s, o, r)*, means that a subject s is permitted to use a right r on an object o. Moreover, the notation *stopped(s, o, r)* means that the access will be revoked because all or some of the decision factors are no longer satisfied. Attributes update can be achieved in UCON before the access (PreUpdate), during the access (OnUpdate) or after the access (PostUpdate) [38].

2.8 Ubiquitous (Pervasive) Computing

2.8.1 Introduction

One of the most rapidly developing areas in ICT is known as 'ubiquitous computing'. It refers to the ever-increasing phenomenon of integrating and embedding ICT tools in people's daily lives and in the situations or environment in which they live. This has been made possible by the ever-improving developments in the manufacture of microprocessors, which now have built-in communication functions and other amenities [77]. There are many applications available for ubiquitous computing, including healthcare, homecare, environmental monitoring, intelligent transport systems management and monitoring, etc. [105]was the first to introduce the term of ubiquitous computing as a seamless integration of micro-devices into the daily lives of ordinary people as well as specialists .

Moreover, developers in the field have been discussing and investigating pervasive or ubiquitous computing since about 1993 [57]. Ubiquitous computing has been defined in many ways but the most broadly used definition refers to it as a method for enhancing the use of computers by users who cannot see them even though there may be many of them accessible in the physical environment. The nature of "environment" in ubiquitous computing differs from that in traditional computing models, wherein the environment is represented as physical space, which is directly supported by the relevant software and hardware to enable interactive messages to be sent between users and that space [48].

2.8.2 Ubiquitous System Properties

In reference to Figure 2.3, there are three main properties in ubiquitous computing (ubicomp) systems, namely, distributed computing, implicit Human Computer Interaction and context-awareness, plus two other additional properties, namely, autonomous and intelligent computing [80].

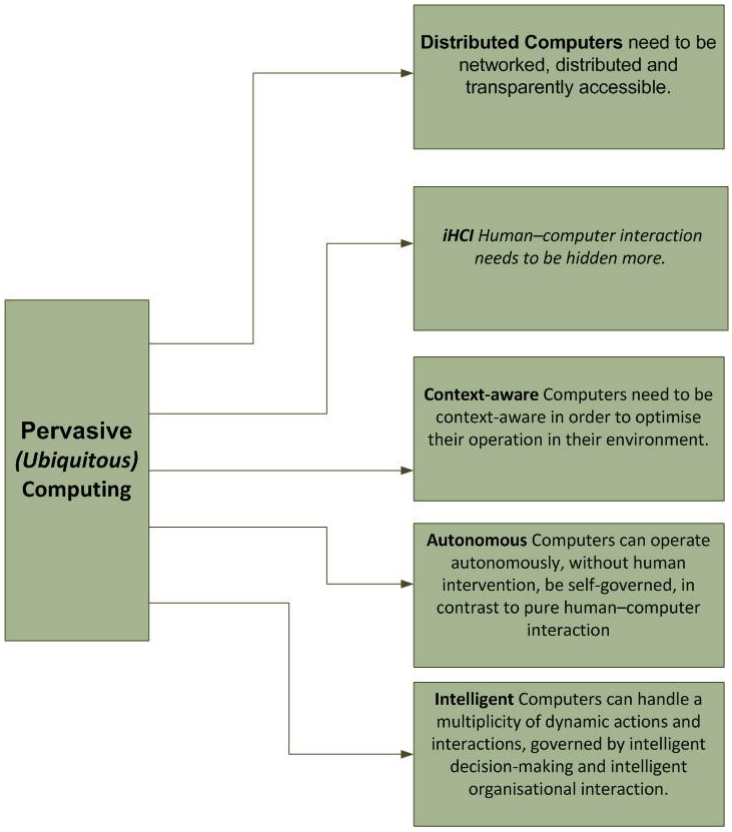


Figure 2.3: Ubiquitous Systems properties [80]

2.8.3 Ubicomp Technologies

Ubiquitous computing involves the convergence of three ICT areas, namely: computing devices, communications and user interfaces [31][8][51].

2.8.3.1 Devices

Personal Communications Services (PCS) devices come in varying shapes and sizes, such as tiny handheld devices as small as mobile devices to near-invisible micro devices that are embedded in everyday objects such as clothing and furniture. These micro-devices have the ability to communicate and interact with each other seamlessly and intelligently. So, devices can be categorised as follows:

Sensors: these serve as a form of input device that function to detect and recognise human commands, and to detect relevant changes in user behaviour as well as in the environment.

Processors: there are functional components for interpreting and analysing the inputs detected by the sensors.

Actuators: these are output devices that act in response to processed information by making certain changes to the environment through mechanical or electronic means, such as controlling air temperature (which is normally done by actuators). Furthermore, the term may refer to devices that deliver processed information instead of being devices that physically change the environment .

2.8.3.2 Connectivity

Ubicomp systems depend largely on the interaction of different independent electronic devices into a larger seamless computer network. The interlinking and interconnectivity of these systems and devices may be achieved though wired (e.g., Ethernet/ADSL broadband) or wireless means (e.g., WiFi, Bluetooth). The devices themselves are equipped with the ability to select the most effective connectivity mode at any given time. The development of ubiquitous computing relies mostly

on the extent of the interoperability and convergence of both wireless and wired technologies.

2.8.3.3 User Interfaces

A user interface is the domain of interaction between a human user and an ICT device. For example, in personal computers, there are various devices, such as a mouse, a keyboard and a microphone, that are used to provide input data, while visual display units, such a monitor or a projector, are used to provide the processed output information. For ubicomp systems, different types of enhanced user interfaces are being developed constantly, with ability to sense and provide data about users and the environment to a computer device for further processing. Interfaces have recently been developed to accept touch as input data, and in the future, user interfaces will be used to accept input data that may be visual, such as face recognition or responses to human gestures; they may include other input types such as sound and scent. Current ubicomp technologies have the potential to know and recognize the user, in the form of the user's expressed attitudes, behaviour or preferences, and subsequently they conduct changes in the physical environment to meet the user's specific needs. However, there are considerable engineering challenges in designing and developing a computing system that can automatically adjust itself, i.e. that can adapt or change as a result of unforeseen situations .

2.9 Overview of Context-Aware Systems

2.9.1 Definition of Term Context

Many researchers have attempted to define the word "context". The first authors who introduced and defined the term context were [93]. They stated the first description of the context as identities of people, location and objects around. [67]

has made similar description of previous one which define context as the users identity, environment, time and location. While [18] have made definition of context as social, user's physical, emotional or state of information. Another way of describe and define the term context is by utilizing synonyms like (situation, Background or circumstance)[14][6].

Moreover, a further definition was posited by [75], who considers context as being the conceptual and physical states of interest to some specified person, place or object. Thus, context is basically information pertaining to the environment, as stated in [81]. Even with all the definitions mentioned above, it remains difficult to clarify the term context; however, arguably the best (and possibly the easiest definition to fully comprehend) was given by [24], who described it as "any information that can be used to characterize 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 application themselves".

2.9.2 Context Model

The importance of the definition of context is that it identifies the precise data that need to be stored, which in turn assists in developing a context model that has the ability to define and store this context information in machine processable form. [98] categorized context models as follows:

1. **Key-value Model:** this is one of the easiest forms for modelling context information. Context information can be stored as attribute-value pairs; For instance, name: John, age: 23, location: office, time: 2:30, and so on. However, this model is limited in terms of being able to refine the structure so that it can handle sufficiently powerful context information retrieval algorithms.

2. **Markup scheme model:** this is another approach to data structure that can be utilized to represent and exchange context information, based on markup tags that are used on the stored data content and attributes. The markup tag contents can recursively contain other markup tags.

3. **Graphical model:** in this model, the unified modelling language (UML) is used because UML has a variety of efficient graphical tools. The generic structure of UML is appropriate for context information modelling. This type of context modelling is suitable for expressing Entity Relationships (ER) from which ER models can be derived from it; these can be used as functional tools to structure relational databases within the system architecture of context management.

4. **Object Oriented Model:** the nature of context information is becoming a problem in most context-aware systems because of their dynamism. Therefore, the goal of using this approach is to encapsulate the details of context processing at the object level and to access the context information only through particular interfaces.

5. **Logic based model:** the descriptions of context in this approach are known as expressions, facts and rules. The expressions also incorporate a set of varieties. All logic-based models employ a high level of formality.

6. **Ontology based model:** the idea behind this type is to identify the context and concepts as well as their interrelations. There are many examples for this approach, but one excellent example that used ontologies to model context information is the CoBrA system (Context Broker Architecture) [21], which characterizes entities by offering a set of ontological concepts.

The most appropriated context model approach for our work in this thesis is object oriented model where The contextual information is embedded as the states of the object, and the object provides methods to access and modify the states.

2.9.3 Context-Aware Systems

Context-awareness is one of many wide areas in ubiquitous computing. Ubicomp technologies allow for separation between users and devices, hence providing anytime, anywhere by anyone computer usage. In order to be effective in the provision of adequate services for users, software applications and services must know their context and have the ability to automatically adapt to changes in those contexts [106]. [6][80] defined a context-aware system as one that has the ability to adapt itself to the current environment or context. So, the system is context-aware if it senses and reacts to changes in its environment. The author who first introduced and defined context-aware computing was [93] in 1994, who defined it as the capability of computing devices to interact with their environment in a dynamic manner. Also [67][75][16] have offered definitions for a context-aware system, generally saying that it reflects the ability of a system to sense or detect context information and to process it, and then to react or respond in order to provide the most appropriate service or information to the user.

In this regard, context-aware applications have been defined by [18] as applications that dynamically deliver services or information, and that react based on any user context provided by sensors.

Context-awareness provides both application developers with new opportunities for collecting context data and then adapting the behaviour of the system accord-

ingly. With the use of mobile devices, context-awareness has become critical to enhancing usability [19]. Although context-awareness has been the subject of much consideration in research and development, there are as yet no specific approaches for the implementation of context-aware systems; this is largely because such systems depend on constraints and situations such as the positions of sensors, the number of probable users, the accessible resources, the typical devices used or the possibility of extending the system [17] [92].

The most appropriated definition for our work in this thesis is defined by [24] who states that a context-aware system is one that has the ability to utilize context in order to deliver suitable services or information to the requester, where the provided service is based both on the user's task and his/her context.

Overall, context-aware systems are designed with the ability to automatically adapt to current context, taking into account the environmental context, without the need for any direct intervention by the user, and thereby to enhance the effectiveness of the device being used [69].

From our previous definitions of context-aware systems, there are two different types of context-aware systems as follows [48]:

- Active context-awareness: a system can sense the context and automatically adapts to new situation based on this context.
- Passive context-awareness: a system can sense the context and presents this context to interested user or makes the context persistent for the user to retrieve later, but cannot adapt to new situation based on this context.

Therefore, in our model we have used an active context-awareness because CA-UCON model can sense the context and adapt to the new situation depend on the

current context. Active context-awareness may help to eliminate user intervention and adapts automatically.

2.9.3.1 Context Information Acquisition

There are various approaches to acquiring context information, as presented by [57]:

1. **Direct sensor access:** this is where client software on the device is able to directly gather the desired information using internally built-in sensors.

2. **Middleware infrastructure:** one important method that modern software design uses is encapsulation; it allows the separation of the graphical user interface from the business logic. Encapsulation as a middleware-based approach that provides a layered architecture model for the context-aware system with the aim of hiding the details of any low-level sensing.

3. **Context server:** in this, the sensors send all the data they gather to the context server for the purpose of facilitating multiple concurrent access. It is logical to allow multiple clients to have access to a remote data source in context-aware systems. The middleware-based architecture is extended by the distributed approach through processing access by managing many distant components.

2.9.3.2 Context-Aware System Abstract Architecture

In this section we discuss conceptual architecture layers for context-aware systems; as illustrated in Figure 2.4 below, expansion layers are designed to detect and for utilizing context information by attaching 'reasoning' and 'interpreting' functionality [57].

The initial layer in the figure above consists of gathering data from the various sensors. It should be noted that 'sensor' here refers not only to sensing the hardware

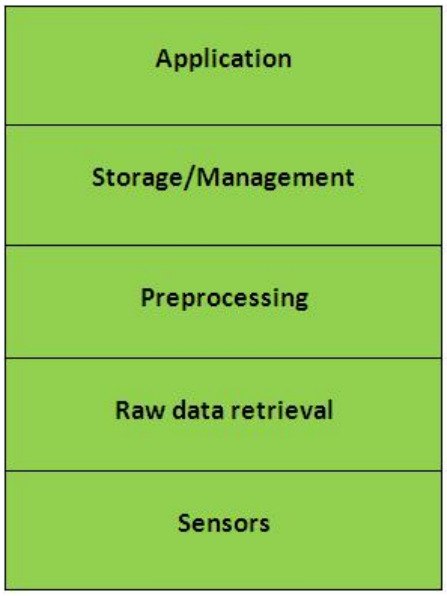


Figure 2.4: Layered conceptual framework for context-aware system [57]

but also all the data sources that are able to offer useful and utilizable context information. In respect of the data capturing methods, sensors may be classified into three groups:

- **Physical sensors:** this is the most commonly and widely used form. There are many varieties of hardware-based sensors that are capable of capturing all types of physical data.
- **Virtual sensors:** these sensors source their context data and information from services and software applications. For instance, determining a particular employee's present location can be achieved by using physical sensors in the form of tracking systems, and also using the virtual sensors in the form of assessing employee browsing information, travel bookings, electronic calendars, emails, etc.
- **Logical sensors:** these sensors utilize of combination of physical and virtual sensors as well as additional information from various sources to resolve higher tasks.

The second layer of this conceptual framework is charged with the recovery of raw context data. Suitable drives are used for physical sensors, while application programming interfaces (APIs) are used for both logical and virtual sensors.

The pre-processing layer is the third one in the framework but is not applied in all context-aware systems. However, it can provide helpful information that can be used in the system if the raw data are too coarse-grained. The pre-processing layer is tasked with performing the reasoning and interpreting of contextual information.

The fourth layer, storage and management, is responsible for organizing the data gathered and subsequently making them available to clients through a public interface. Clients can access information in two ways: the first is access through a synchronous manner, whereby clients poll the new changes on the server using a remote method. The second is asynchronous, whereby clients subscribe for a particular content or event in which they are interested.

The fifth layer in framework is application. The implementation of the real response to various context information and events is achieved here. In some circumstances, application-specific context, reasoning and information retrieval management are encapsulated and hidden from agents, which act between the pre-processing and the application layers as an additional layer in order to communicate with the context server.

2.10 Adaptive Systems

From the literature, there are a variety of definitions for adaptive system; one of the most common definitions was given by [82], which states, "Self-adaptive software

evaluates its own behaviour and changes behaviour when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible". The appropriate definition for our research was given by [36],which states, "Self-adaptive software modifies its own behaviour in response to changes in its operating environment. By operating environment, we mean anything observable by the software system, such as end-user input, external hardware devices and sensors, or program instrumentation" . The abstract architecture of adaptive systems can be seen in Figure 2.5 .

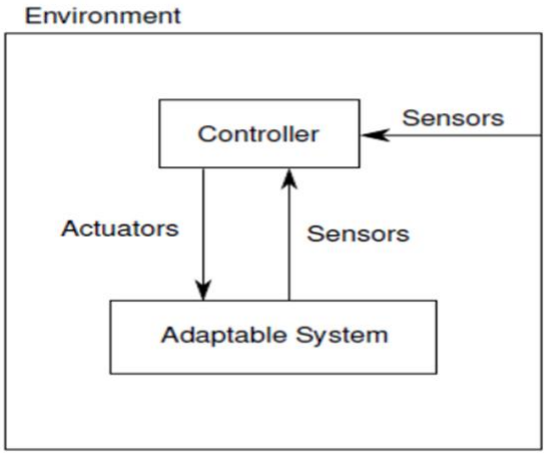


Figure 2.5: Adaptive System Architecture [82]

There are different kinds of systems that have link and similar meanings to the term adaptive system, including as self-managing and autonomic system. However, it is challenging to identify any real differences among these terms, and they are used interchangeably by the majority of researchers. Nevertheless, the distinction between autonomic and self-adaptive is that the former is more general, whereas the latter is more restricted, which means that an adaptive system is a special case of an autonomic system.

2.11 Adaptation in Ubiquitous Computing

Ubiquitous computing adaptation is considered as a reactive process, in which the adaptation takes place based on a particular circumstance or series of events in the environmental context; the main objective of this adaptation is to improve the quality of the service being used by the system user. Therefore, the most significant requirements for an application being applied in the ubiquitous environment are that it has the ability to sense the surrounding environment and process these changes, and that it can respond to these changes in an efficient and effective manner [43].

The most common description for adaptation within ubiquitous computing is taken from the MAPE-K loop developed by IBM [25], which is often used in the autonomic computing context. Thus, adaptation in ubiquitous computing is considered as a closed loop consisting of different phases, as shown in Figure 2.6.

The first phase is the sensing and processing of context; this phase senses the different user contexts, such as location, user preferences, etc., and the different system contexts, such as light level, temperature, etc. All the gathered data are interpreted in terms of high-level context events in order to consider the various adaptation process steps available.

The next phase is reasoning and planning; the function of this phase is to process and analyse any new changes to the context that have been captured by the sensors, to consider the particular type of adaptation required as well as how to accomplish the required objectives.

The final phase in ubiquitous adaptation is adaptation acting; the responsibility of this phase is to implement the most suitable adaptation approach in order to effect

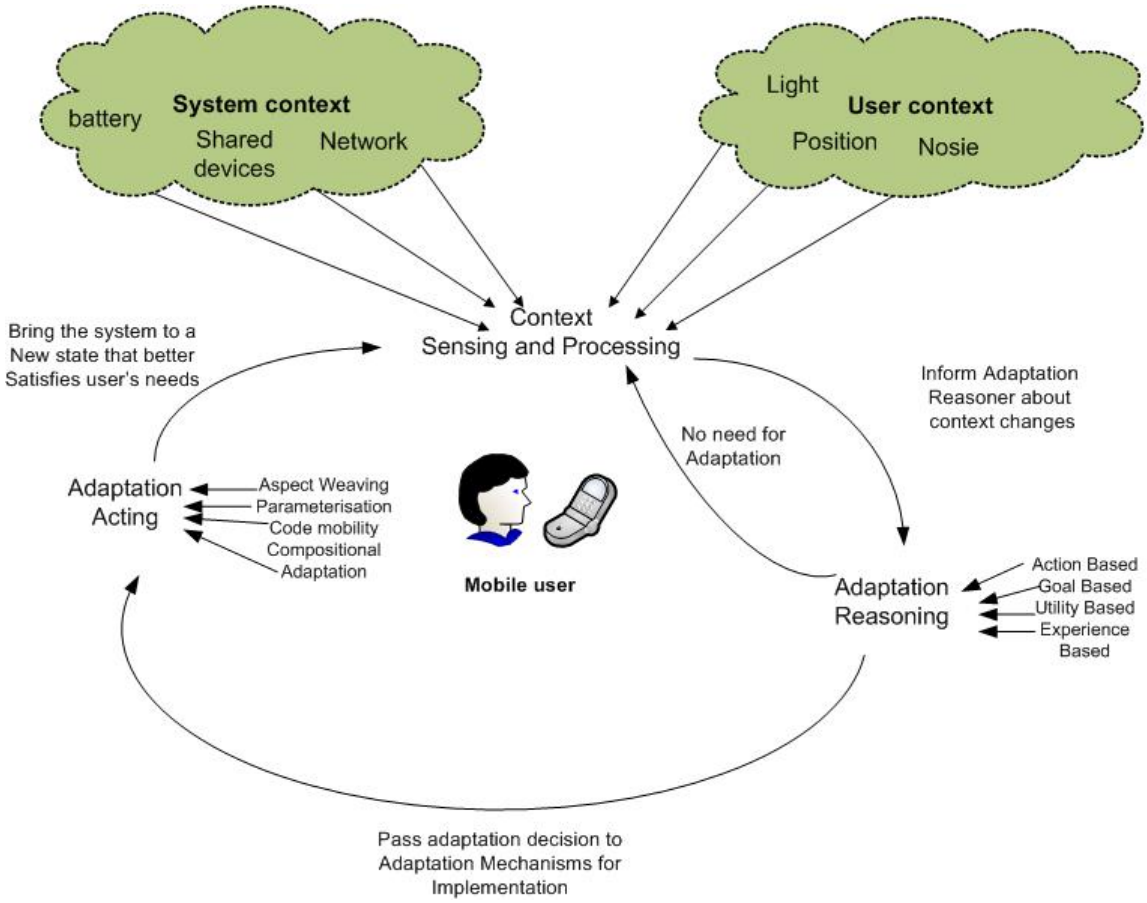


Figure 2.6: Adaptation loop in ubiquitous computing [25]

the adaptation decision of the previous phase (reasoning and planning process).

2.12 Adaptation Approaches

From the literature, various adaptation approaches have been utilized in order to realize dynamic adaptation in software. In the following subsections, we present three adaptation approaches, known as parameter adaptation, compositional adaptation and action-based adaptation [63][20].

2.12.1 Parameter Adaptation

This kind of adaptation approach is utilized to affect the behaviour of the system via the amendment of particular variables in the program. A common instance of this approach is TCP protocol, in which its behaviour can be adjusted in respect of network congestion. Parameter adaptation is used in the area of ubiquitous computing for amending the non-functional properties of a system that are influenced as a consequence of context change. For instance, a portable device that has an image render might present images in low quality because of the low bandwidth or low memory size, and so the system adapts its behaviour to the current situation by adjusting particular parameters in the image application in order to meet the new situation [30].

However, having unpredictable context changes in a highly dynamic environment raises the difficulty of trying to define in advance all the probable contexts (as well as their processes) for non-functional properties. The advantage of the parameter adaptation approach is that it is cheap in terms of implementation effort and complexity, and this is why some context-aware systems use this approach [4].

On the other hand, the parameter adaptation approach has not been considered as an optimal solution in the area of ubiquitous systems; the key disadvantage of this approach is that the software components (as well as unimplemented algorithms) that are left throughout the design stage cannot be adapted. In addition, using this approach for such an application, where its behaviour is based on frequent context changes, might lead to several configurable parameters becoming conflicted. Thus, a different approach is required, i.e. one that helps to decrease the number of parameters; like memory parameter. [7].

2.12.2 Compositional Adaptation

Another approach to adaptation is a compositional adaptation; this goes further than straightforward code-tuning and permits the algorithm or parts of the system structure to be replaced to enhance the program in order to meet the current situation. Compositional adaptation works in environments with unpredictable contexts or requirements and where new adaptation functionality may be required. Thus, in the following two bullets, we discuss the most important technologies that have been used in compositional adaptation [63][11]:

- Separation of Concerns: in this technology, the segregation of functional behaviour from non-functional behaviour in terms of development is permitted, where the functional behaviour is related to business logic and the non-functional behaviour is associated with security and quality of service. This mechanism facilitates the development of the system and its maintenance when upgrading the system. Separation of concerns, such as domain-specific languages, constraint languages and generic languages, has been considered a significant principle in the area of software engineering, and it is employed in various advanced techniques .

- Computational Reflection: this is related to the ability of an application to process new contexts and adapt its behaviour to the current situation. It facilitates a system in changing its behaviour through expressing the implementation details of the system at an abstract level, devoid of negotiating portability. In this technology, two activities are introduced: introspection and intercession; the former deals with the observation of the system behaviour, and the latter responds to the changes that are captured in the observation stage and then adapts to the new situation.

2.12.3 Action-Based Adaptation

This kind of adaptation approach is also known as rule-based adaptation. Action-based adaptation is a common approach employed in adaptive systems. It is able to define the self-configuring and self-managing aspects of the system's behaviour that is associated with distributive technologies and networks. This approach relies on the concepts of state and action, where the system at a specific time (t) should move from the current state (S1) to the next state (S2) if all corresponding conditions are true and the transition of the system has to be determined by the action (a). Therefore, the way to direct the system to adapt to the new situation is through using the format of If (conditions), and Then (actions)[35] [70].

Many researchers have used this type of adaptation approach in their work. One common model was proposed by [27], who developed an adaptation platform for mobile systems; it uses action policies based on an event calculus, which are formulated in the form of conditions and actions that accurately determine the adaptation behaviour of the system. The conditions are defined as logical expressions that may take the value "true" or "false", while the actions are defined as the adaptation methods that are performed if the condition is evaluated to be "false".

Moreover, the notion of event-action rules has been used in terms of expressing dynamic system reconfiguration. For instance, the DART project [46] uses an adaptation manger that implements adaptation policies, which are activated by particular events, and these events are created depending on user requirements and system statistics. Thus, each policy is associated with one or more events, and whenever a specific event occurs, the most suitable policies are invoked by the manager, who then executes them. The problem is that many policies might index the same event,

and this could cause the policies to conflict; in order to solve this problem, appropriate priorities are allocated to each policy.

2.13 Comparison of Adaptation Approaches

In this section, we present the differences between all the adaptation approaches mentioned above in order to identify the one best suited to the field of ubiquitous computing. The parameter adaptation approach is suitable for a low-dynamic environment; it can change the system's behaviour by alerting particular parameters that are influenced by context changes. However, this adaptation approach is not appropriate approach for highly dynamic environments, which must consider all the various types of context that could occur and respond based on those contextual changes. The second adaptation approach is compositional; this is considered to be a general approach to self-adaptation based on the language of implementation. Depending on several works that have been accomplished through compositional adaptation, it can be claimed that compositional adaptation is a powerful approach for two different reasons; the first, in terms of processing level, is its flexibility of reasoning, and the second, in terms of adaptation reaction, is the easy implementation of the dynamic reconfiguration of the system (compositional adaptation allows algorithmic and structural changes) [63].

However, the action-based adaptation approach is considered a good alternative solution; it uses actions in order to adapt to new situations. This type of adaptation approach is more suitable for non-functional requirements such as security and quality of service. Based on our adaptation objective, we would prefer to use this type of adaptation approach, as the CA-UCON model uses adaptation actions in adapting

to the current context. The action-based adaptation approach uses conditions and actions, where the actions take place if the conditions are evaluated as being false in the adaptation process.

2.14 Related Work on Context-Aware Access Control Models

In this section, we present some additional works that have been done in the area of context-aware access control models, which merge the context information with credentials while making any decision over access control. In other words, the context-aware access control model is an access control mechanism that uses context information as a further constraint to govern and control the resources or adapts to the new situation [34].

2.14.1 Extensions of Role-Based Access Control (RBAC)Model

In this subsection we present some proposed models that extend the RBAC model to include the context information in the access decision. We, then compare these models with our proposed model in order to show the differences and the limitations.

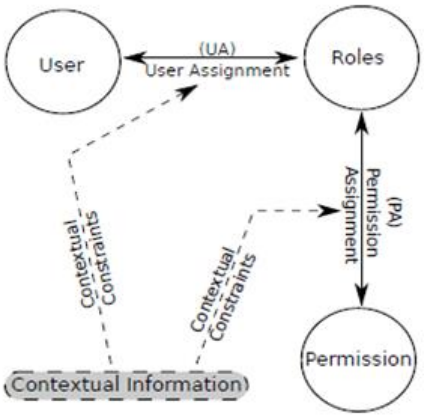


Figure 2.7: CAAC conceptual view [26]

As can be seen in Figure 2.7, the new kind of restriction introduced to the context-aware access control model is known as a contextual constraint, which controls the user assignment and permission assignment functions [26]. For example, a specific role is granted to a specified user only if the contextual constraints of that user are fulfilled, and permission is assigned to that role only if the contextual constraints of the holder of this role are fulfilled. A further discussion of such models is given in the following subsections.

2.14.1.1 Generalised Role-Based Access Control Model

This is one of the earliest models and was proposed by [29] in the area of context-aware access control. This model was proposed for the *smart home* environment, which extends the notion of the *role* to include two kinds of roles, which are known as object role and environment role. The environment role is utilized to capture the environment context, which means the system context as well, as it is considered in the decision-making process. However, the object role is used to capture the level of sensitivity of the requested object. For instance, if a subject wants to perform a particular right over a particular object, the access control considers the object role, subject role, environment role and the right in order to permit the access. The purpose of these widened roles is to eliminate the restriction of the subject-centric roles of the RBAC model. Because of the introduction of these extended roles, the GRBAC model becomes more complex in term of decision-making, which means that it requires a complex framework as well [45][47].

2.14.1.2 Spatio-Temporal Models

Several access control models were developed in the area of pervasive systems, and in order to control the access, they generally use two types of context information: time and location [99] [101]. One of these models is known as Temporal Role-Based

Access Control (TRBAC) model proposed by [10], which is an extension of the RBAC model. This model introduced a new notion called *temporal restriction* within access control policy, which offers a new mechanism that implements the access control policies based on time. Therefore, the assignment of any role to the user relies on satisfying the temporal constraint. Another model is known as generalized Temporal Role-Based Access Control (GTRBAC) model [41], which extends the above model (TRBAC) by including the idea of an *activating role*. The difference between an activating role and an enabling role is that an enabling role can be claimed by the user with all the related permissions. However, an activating role is an enabling role that must be activated in the particular session by at least one user. So, the user can obtain all the related permissions of the current role. The advantage of the activating role is that it assists in identifying the currently running role, which is utilized to observe all running activities and usage of resources. Many temporal restrictions are enforced by the GTRBAC model on user assignment and permission assignment policy, enabling time and role activation [22].

A different model was proposed by [109], which is known as Spatial Role-Based Access Control (SRBAC) Model. This model considers a constraint that is based on location. In this, the access permission is granted to the user based on the user's location; the location may be divided into several zones and each one may be assigned to a different set of permissions. Thus, if the user requests access, the condition of this role must be fulfilled in the current zone. From the table 2.1, it can be seen that each user role is assigned with a different permission in a specific location. The main problem of this model is that it does not have the proper semantic meaning of the position information.

Various models have been categorized as spatio-temporal access control methods, which consider location and time in terms of contextual information in the access decision process, including those explained in [100] [49].

Table 2.1: Location Permission Assignment List in SRBAC

Roles	Location	Permissions
Customer_role	Zone1	P1,P2,P3
Customer_role	Zone2	P4
Customer_role	Zone3	ϕ

2.14.1.3 Dynamic Role-Based Access Control Model

The models examined above have a common aspect in that they consider the context information of the user at the time of the request. Thus, the user's context information is evaluated, which means that if the user requests to access an object, the system evaluates the context information of the user in order to determine whether or not to allow access to the requested object. The evaluation of the context information is performed at the time of the request, and there is no further evaluation once the right has been granted to the user. Moreover, only time and location as context information are considered in spatio-temporal models, even though context information is represented by more than just time and location. Therefore, in the pervasive environment, all types of context information should be considered in order to build a comprehensive access control model.

To solve this problem, a model was proposed by [108], which is known as the Dynamic Role-Based Access Control (DRBAC) model. In this model, the role and permission assignments are regulated dynamically, relying on the context information of the user, but it considers all types of context information by using the Central Authority (CA) in order to deal with and manage the role hierarchy for the user in granting those roles. In the CA, an agent is utilized in the device of the user in order to observe and alter the role(s), based on the context information. The agent is represented by a state machine, which means that each state represents a role, and the transition between the current role to another role is performed by moving from the current state to another state; this happens based only on the context

information.

The key disadvantage of the DRBAC model is that it is highly complex in terms of implementation, as the user device must have a role state machine running for each role. This increases the number of roles in the system, which increases the complexity of the system; this is a major problem for the restricted devices that are generally used in pervasive environments.

Various models have been categorized as Dynamic access control methods including those explained in [15][115]

2.14.1.4 CAAC-based Models with Architectural Components

Context-aware access control models require a special architecture to gather and manage the context information that must be considered in the decision-making process. This architecture must contain two parts: one for the collection and management of the context information, and the second one for controlling the access. However, the architecture that represents the traditional access control models includes only one part, which is access control, and so is not suitable for models in which context information is considered in decision-making. In order to conquer this restriction, a variety of architectures have been proposed recently to capture the context information. In the following paragraph, we present one example of these architectures [44].

UbiCOSM: One of the context-aware access control models is known as UbiCOSM [23]. It uses the evaluation of the user's context in order to control the objects or resources. As can be seen from the figure 2.8, the UbiCOSM model includes the external model that manages the context and that deals with low-level entity identification (like CARMEN [9]). It is hard for UbiCOSM to hold diverse access control requirements, application domains and policies because of the strong

connection between context information and access permissions.

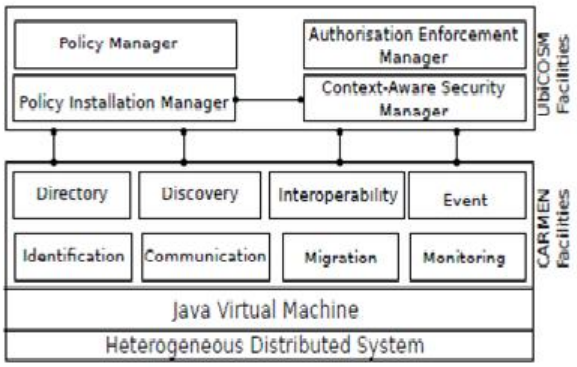


Figure 2.8: UbiCOSM middleware services [23]

The reason behind introducing the authorisation enforcement manager in UbiCOSM is to segregate the authorisation engine from the access control enforcement. This separation offers a high degree of modularity, while the enforcement of an access control is based on application.

However, The majority of these works and their solutions have been constructed on the RBAC model. All the above RBAC extensions are simply represented in CA-UCON model, where CA-UCON has the preAdapting components that can adapt to the new situation before the access request is granted (based on the context information). In addation, CA-UCON goes beyond that by controlling the usage and adapt to the change in the environment in order to preserve continuity of usage. So, each extension of traditional access control above is considered as a special case of CA-UCON model.

2.14.2 Extensions of Usage Control (UCON) Model

Many recent works have been done to improve the UCON model and solve this weakness in pervasive computing system. For instance, [113] proposed a new access control called as Times-based Usage Control (TUCON) in order to protect digital resources misuse. In TUCON the use of the time variable is Initiated into UCON,

and greatest times described as consumption restrictions. This approach is simply identified in CA-UCON model by specified the time as condition requirement. [39] proposed a new model defined as Geography Usage Control (GEO-UCON) model to deal with GEO DBMS access control. In this model, a geospatial factor is introduced into UCON in order to ensure data protection in mobile applications and location-based services. This model like the last one, it can be defined in CA-UCON model where the location can be used as condition requirement to control the service.

Moreover, [5] proposed usage control model to context-aware in mobile computing environments defined as ConUCON. In this model two new components are introduced: context and states. The access decisions in ConUCON model is based on these new components plus obligations. [58] proposed a new model called as contextual Usage Control (CUC) model which replaces the conditions component in UCON by context and add a management module to it. The last two models are using context information as further constraints in order to govern and control the resources by changing the condition component into context, which is basically the UCON model can do that. UCON model has condition component which can sense the context of environment, but it cannot adapt to the new situation based on this context. However, all these efforts have not completely solve this issue.

In contrary the above UCON extensions, CA-UCON model enables adaptation to environmental changes in the aim of preserving continuity of access by triggering specific actions to adapt to new situations. In addition to data protection, CA-UCON model enhances the quality of services, striving to keep explicit interactions with the user at a minimum. This makes it more suitable for pervasive computing

systems.

2.15 Summary

In this chapter, we presented a background about traditional access control models (MAC, DAC,RBAC), trust management(TM) and DRM by discussing the purposes, functionality and distinction between these models. We then, investigated usage control model (UCON) which is considered as a new access control model and determines what distinguishes it from other traditional access control models.

In next part of this chapter, we gave an overview of ubiquitous systems and context-aware systems by examining the context and context-aware system and properties of Ubicom systems. In addition, the definition of adaptive systems and the adaptation in ubiquitous system are presented. Then, a various aspects of adaptation in term of adaptation approaches are covered. Finally, previous works on context-aware access control are provided.

In the following chapter, we propose a context-aware and adaptive usage control model by demonstrating architecture and computational model of this model.

Chapter 3

Context-Aware and Adaptive Usage Control (CA-UCON) Model

Objectives:

- Present a novel architecture of the CA-UCON Model.
- Present a computational Model of CA-UCON .
- Give formal definition of CA-UCON model .
- Show the Expressive power of the CA-UCON model.

3.1 Introduction

In this chapter we propose a Context-Aware and Adaptive Usage CONtrol (CA-UCON) model which extends the traditional UCON model to enable adaptation to environmental changes in the aim of preserving continuity of access. Indeed, when the authorisations and obligations requirements are met by the subject and the object, and the conditions requirements fail due to changes in the environment or the system context, CA-UCON model triggers specific actions to adapt to the new situation. Besides the data protection, CA-UCON model so enhances the quality of services, striving to keep explicit interactions with the user at a minimum. We, then propose a novel architecture for Context-Aware and Adaptive Usage Control (CA-UCON) model which is introduced based on the notion of context-aware system. The main innovative features of this architecture is the merging of continuity of usage decision (UD) and dynamic adaptation decision (AD) to changes in the environmental or system context. The computational model of CA-UCON is described as a Finite State Machine (FSM), depicting how a subject's request to access an object is handled in this model. Moreover, we provide the formal definition of the two newly introduced adaptation models within the CA-UCON model. The expressive power of CA-UCON model is presented which the UCON model can be specified in CA-UCON model; and so all the access control models that can be represented in UCON (such as RBAC, MAC, DAC, DRM).

3.2 Architecture of CA-UCON model

In this section, we elucidate the whole architecture of the CA-UCON model as depicted in Figuer 3.1, it provides comprehensive definition of each components in the architecture and reveals the functions performed by each components in the archi-

tecture, in particular how these components interrelate with each other to complete the task of permit or deny the access request. The architecture highlights the two important components known as usage decision (UD) component and the adaptation decision (AD) component. The two dashed ovals materialise the fact that usage decision and adaptation decision happen continuously before and during usage. The following subsections explain the intuitive meanings of the two significant components with all corresponding components.

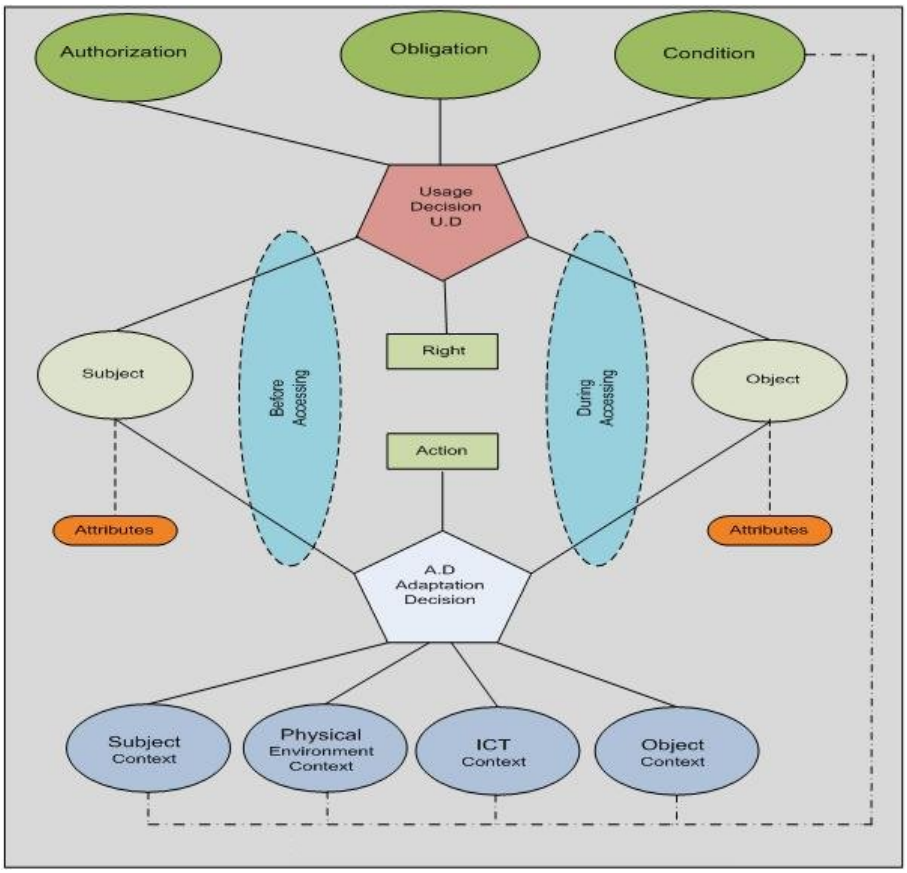


Figure 3.1: Architecture of The CA-UCON model

3.2.1 Usage Decision (UD)

This component is described as in the usage control model (UCON), making decision of granting or denying rights of access based on three factors identified as

authorisation, obligation and condition components [73]:

- **Authorisations (A):** Authorisations are a key functional requirement that must be fulfilled before granting a particular right of access to a digital object. Authorization predicates place conditions and constraints in the form of logical predicates on both the subject and object attributes. The authorization predicates are activated and evaluated both before (pre-authorization) and during (on-authorization) access.

- **oBligations (B):** Obligations are also functional predicates which are used to confirm mandatory requirements that a subject must undertake both before and during a particular usage process. The mandatory requirements here may be either pre-obligations (preB) to be fulfilled before access permission is granted or on-obligations (onB) to be fulfilled during access.

- **Conditions (C):** Conditions are environmental constraints that must be considered in the process of usage decisions. Conditions are not related directly to objects or subjects, but they are based on environmental attributes. The evaluation of condition predicates may take place before granting permission to access a digital object (pre-conditions) or while the subject is using the object (on-conditions). When conditions fail due to changes in the environmental context, adaptation actions are triggered in an attempt to change the environmental context such that these conditions hold.

3.2.2 Adaptation Decision (AD)

This component decides what adaptation action to perform depending on the environmental context which is refined into subject context, object context, information and communication technology (ICT) context, and the physical environment context:

- **Subject Context:**

  Subject context is any type of context information linked to the subject such as his location, activity, preferences, and people nearby.

- **Object Context:**

  Object context refers to any kinds of context information related to the object. These can be the location of the object, execution state, nearby resources and availability.

- **Physical environments context:**

  This characterises relevant physical phenomena taking place such as the time, light, noise level, temperature, weather and so on.

- **ICT context:**

  ICT context is general term that deals with any kind of context information related to ICT and computing system included any communication devices or applications nearby. Examples of these contexts include: laptops battery rate, network reliability, smart phones memory size, PDAs and hardware capability and communication bandwidth. In addition, the diverse services and applications related to them, such as video-conferencing and distance learning.

- **Adaptation Actions:**

  Adaptation action is an operation that should be performed over condition predicate with the purpose of overcoming the environments changes. These actions may be classified according to the subject of the adaptation and the scope. For example, service instance adaptation actions (retry, duplicate service, and substitute service) and flow instance adaptation actions (redo, choose alternative service, and undo).

3.2.3 Subjects (S) and subject Attributes (ATT(S))

A subject is an entity who requests access to a resource and must hold certain rights of access to the target object or resource. Subject has attributes which are used in the usage decision making. For example of subject attributes are : identities, roles, group name and memberships. We let S denote the set of subjects and ATT(S) denote the set of subject attributes.

3.2.4 Object (O) and Object Attributes (ATT (O))

Object is the resource or entity which the subject has to hold a certain right to access or use. Object attributes are the descriptions and properties of a given object which could be used as the basis for the provision and making of the usage decision process. For instance of object attributes are: ownership, security labels and role permission. Let O denote the set of objects and ATT(O), the set of object attributes.

3.2.5 Rights (R)

Rights are privileges that subject can hold and use on an object. The subject must fulfill the authorizations, obligations and conditions requirements in order to be granted the right to access the object. For example of rights are: *read, write and download* The subject loses this right anytime one of these requirements does not hold. If this happen during access, there are two possibilities: (i) if either authorisations requirement or obligations requirement is not fulfilled, then the access right is revoked and the access stopped at once; (ii) if both authorisations requirement and obligations requirement are fulfilled, but the conditions requirement is not met due to changes in the environmental context, the system will attempt to adapt to the new situation by performing specific adaptation actions (including request to alternative object); if the adaptation is successful then the access continues, otherwise

the access right is revoked and the access terminated.

3.3 Computational model of the CA-UCON model

In this section, a computational model of CA-UCON model is presented , it can be described as a Finite State Machine (FSM) depicting how an subject's request to access an object is handled in the CA-UCON model. The FSM is depicted by the graph in Figuer 3.2, where nodes are called states and edges are called transitions. The initial state, labelled *initial*, corresponds to the state when the system is waiting for a subject to submit a request. There are three final states: *end*, when the access has successfully terminated; *denied*, when the access request has been denied; and *revoked*, when access permission has been revoked during access and hence the access stopped.

The intuitive meaning of the remaining states of the FSM can be summarised as follows: *requesting*, denotes when the access request is being processed; *accessing*, represents the state when the actual access is taking place; *preadapting*, is the state when the system is trying to adapt to the environmental context prior to access; and finally *onadapting*, is when the system is trying to adapt to the environmental context during access.

The transitions of the FSM are labelled with the events (or actions) that fire them. The event *tryaccess* occurs when a subject sends an access request (e.g. by clicking a menu button). This event forces the FSM to enter the *requesting* state to process that access request. While in this state, the system can perform updates on subject's and object's attributes through *preupdate* events. If the authorisations, obligations and conditions requirements are all met, the system emits the *permitaccess* event and moves into the *accessing* state. If for some reasons either the

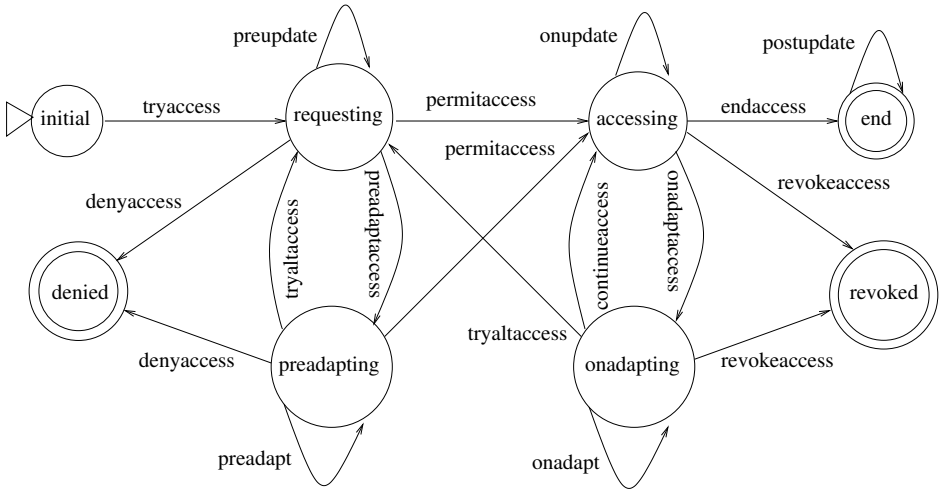


Figure 3.2: Execution of an access request in the CA-UCON model

authorisations requirement or the obligations requirement is not met, the system emits the event *denyaccess* and terminates in the *denied* state. However, if both the authorisations requirement and obligations requirement are met, but the conditions requirement is not satisfied, the system emits the *preadaptaccess* event and moves into the *preadapting* state.

In this state, specific adaptation actions, denoted by the *preadapt* events, are performed in an attempt to meet the conditions requirement. If the adaptation is successful, the *permitaccess* event is raised and the system transitions into the *accessing* state. In addition, a new request to access a specified alternative object, denoted by the *tryaltaccess* event, may be issued automatically by the system if the adaptation actions fail. Otherwise the access request is simply denied when no adaptation is possible.

When access permission is granted (see *permitaccess* event), the system transitions into the *accessing* state in which the actual access takes place. During access the system can perform updates on subject's and object's attributes via *onupdate* events.

If during access either the authorisations requirement or the obligations requirement is not met, the system emits the event *revokeaccess* and terminates in the *revoked* state. However, if both the authorisations requirement and obligations requirement are continuously met, but the conditions requirement fails, the system raises the *on-adaptaccess* event and moves into the *onadapting* state.

In this state, specific adaptation actions, denoted by the *on-adapt* events, are performed in an attempt to meet the conditions requirement. If the adaptation is successful, the *continueaccess* event is raised and the system moves back into the *accessing* state. In the effort to enhance the quality of service even further, the system might issue an implicit request to access a specified alternative object through the *tryaltaccess* event, when the adaptation actions fail. In the worst case when no adaptation is possible, the access permission is simply revoked and the access stopped at once. When an access terminates successully via the *endaccess* event, the system moves into the *end* state and eventually performs updates on subject's and object's attributes through *postupdate* events.
Therefore, The Finite State Machine (FSM) of CA-UCON model in figure 3.2 will be specified using Calculus of Context-Aware Ambient(CCA) and later used to analyse the behaviour of the model.

3.4 The CA-UCON$_{\mathbf{ABD}}$ Family Core Models

In this section, a formal definition of CA-UCON model is presented which extends and enhances the traditional UCON model by enabling the adaptation features on it. [73] defined the UCON$_{\mathrm{ABC}}$ family core models where A stands for Authorisations, B for oBligations and C for Conditions. Here we define the CA-UCON$_{\mathrm{ABD}}$ family core models where C is replaced by D for aDaptation. So the CA-UCON$_{\mathrm{A}}$ and

CA-UCON$_{\mathrm{B}}$ family core models are identical to UCON$_{\mathrm{A}}$ and UCON$_{\mathrm{B}}$, respectively. The CA-UCON$_{\mathrm{D}}$ family core model comprises two new models: the pre-adaptation model CA-UCON$_{\mathrm{preD}}$ and the ongoing adaptation model CA-UCON$_{\mathrm{onD}}$. The CA-UCON$_{\mathrm{ABD}}$ family core models are detailed below.

3.4.1 The CA-UCON$_{preA}$ Model

The CA-UCON$_{preA}$ Model has exactly the same meaning as on UCON Model which evaluates the attributes of subject and object for the decision making before the subject access the object (at the time of access request). So, if the pre-authorization evaluate to true the access request is granted to the subject.

The CA-UCON$_{preA}$ core model is composed of:

- S: set of subjects, *ATT(S)*: set of subject attributes.

  The subject who request an access to an object.

- O: set of Objects, *ATT(O)*: set of object attributes.

  The object is an service or resource that the subject request access on it.

- R: set of rights.

  The right are privileges that an subject has to hold in order to access an object.

- $PreA$: pre-authorizations.

- $allowed(s, o, r) \Rightarrow preA((ATT(s), ATT(o), r)$.

  This predicate denotes that the subject s is allowed to access the object o with the right r, if the predicate $preA$ is evaluated to true.

3.4.2 The CA-UCON$_{OnA}$ Model

The CA-UCON$_{OnA}$ model is also has the same concept like in UCON model which evaluates the attributes to control the usage access during the access time. In

absence of pre-authorization, the requested access is always allowed. However, ongoing authorization is active throughout the usage of the requested right. The CA-UCON$_{OnA}$ core model is composed of:

- S: set of subjects, *ATT(S)*: set of subject attributes.
- O: set of Objects, *ATT (O)*: set of object attributes.
- R: set of rights.
- $allowed(s, o, r) \Rightarrow true$;

  This predicate denotes that there are no pre-authorizations required at the time of access request.
- OnA: ongoing-authorizations.
- $Stopped(s, o, r) \Leftarrow \neg onA(ATT(s), ATT(o), r)$

  This predicate denotes that the right r used by the subject s on the object o is revoked, if the predicate onA is evaluated to false.

3.4.3 The CA-UCON$_{preB}$ Model

The CA-UCON$_{preB}$ Model has defined precisely as the same as in UCON model which is a kind of history function that checks whether certain obligations have been fulfilled or not and return true or false before the access is granted. The *preB* predicate is evaluated to true, if all the required pre-obligation elements *preOBL* are fulfilled using preFulfilled predicate. The CA-UCON$_{preB}$ core model is composed of:

- S: set of subjects, *ATT(S)*: set of subject attributes.
- O: set of Objects, *ATT (O)*: set of object attributes.
- R: set of rights.

- OBS: set of obligation subject, OBO: set of obligation object, OB: set of obligation actions.

  There are three different types of obligations : obligation that are related to the subject which have to be fulfilled by the subject before the access right is granted, obligations that are related to object which must to be satisfied at the time of access of request; obligations that are linked to actions.

- $preB$: pre-obligation predicates.

- $preOBL$: pre-obligation elements

  This predicate is responsible for holding all obligations types.

- $preOBL \subseteq OBS \times OBO \times OB$.

- $prefulfilled : OBS \times OBO \times OB \rightarrow \{true, false\}$.

  This predicate is checking all types of obilgations and return with ture or false.

- $getPreOBL : S \times O \times R \rightarrow 2^{preOBL}$.

  This is a function to select pre-obligations for a requested usage.

- $$PreB(s,o,r) = \left\{ \begin{array}{l} \bigwedge_{(obsi,oboi,obi) \in getpreOBL(s,o,r)} prefulfilled(obsi, oboi, obi) \\ if\ getPreOBL(s,o,r) \neq \emptyset; \\ True\ if\ getPreOBL(s,o,r) = \emptyset; \end{array} \right\}$$

This function denotes that the predicate $preB$ is true, if the prefulfilled predicate returns true or there are no required obligations for current access request.

- $allowed(s,o,r) \Rightarrow preB(s,o,r)$;

This function denotes that the subject s is granted the right r to access the object o, if the predicate $preB$ is true.

3.4.4 The CA-UCON$_{onB}$ Model

The CA-UCON$_{onB}$ Model as the same as in UCON model which is a kind of history function that checks whether certain obligations have been fulfilled or not and return true or false during the access. The *onB* predicate is evaluated to true, if all the required on-obligation elements *onOBL* are fulfilled using onFulfilled predicate. The CA-UCON$_{onB}$ core model is composed of:

- S: set of subjects, ATT(S): set of subject attributes.
- O: set of Objects, ATT (O): set of object attributes.
- R: set of rights.
- OBS: set of obligation subject, OBO: set of obligation object, OB: set of obligation action.
- T: set of time or event elements,$onOBL$: ongoing-obligation elements
- onB: ongoing-obligations predicates.
- $onOBL \subseteq OBS \times OBO \times OB \times T$.
- $getOnOBL : S \times O \times R \rightarrow 2^{onOBL}$, a function to select ongoing-obligations for a requested usage.
- $onFulfilled : OBS \times OBO \times OB \times T \rightarrow \{true, false\}$;
- $onB(s, o, r) = \wedge_{(obsi,oboi,obi,ti) \in getOnOBL(s,o,r)} onFulfilled(obsi, oboi, obi, ti)$;
- $onB(s, o, r) = true$ by definition if $getOnOBL(s, o, r) = \emptyset$
  This predicate denotes that the predicate onB is true, if there are no required obligations during the access.

- $allowed(s, o, r) \Rightarrow true$;

  This predicate denotes that there are no pre-obligations required at the time of access request.

- $stopped(s, o, r) \Leftarrow \neg onB(s, o, r)$.

  This predicate denotes that the right r used by the subject s on the object o is revoked, if the the predicate onB is evaluated to false.

3.4.5 The CA-UCON$_{preD}$ Model

In the CA-UCON$_{preD}$ model, adaptation can be activated only before the access permission is granted. That is adaptation cannot take place during access. If s is subject, o is object and r an access right, we let $preD(s, o, r)$ denote a predicate which is true if the pre-adaptation is successful and false otherwise. We also denote the access permission decision by the predicate $allowed(s, o, r)$. The CA-UCON$_{preD}$ core model is composed of the following elements:

- S: set of subjects, *ATT(S)*: set of subject attributes.
- O: set of objects, *ATT (O)*: set of object attributes.
- AD: set of adaptation actions.
- $PreCON$: set of pre-conditions elements.
- T: time domain.
- $PreAdapted : 2^{preCON} \times AD \times T \longrightarrow \{true, false\}$

  preAdapted(c, a, t) is a boolean function that performs the adaptation action a until all the conditions in c evaluate to true, in which case the function returns *true*; otherwise the function returns *false* after t time-units have elapsed since the execution of the action a started.

- $getPreADAPT : S \times O \times R \longrightarrow 2^{preCON} \times AD \times T$

  getPreADAPT(s, o, r) returns a tuple *(c, a, t)* where c is the set of all pre-conditions required to grant the subject s the access right r upon the object o, a is the adaptation action to be performed if any of the pre-conditions does not hold, and t is the time-out for this adaptation process.

- $getPreAltReq : S \times O \times R \longrightarrow 2^{O \times R}$

  getPreAltReq(s, o, r) denotes the set of alternative requests that can be made on behalf of the subject s when the initial request of the access right r upon the object o could not be granted due to environmental conditions.

- *preD(s, o, r) = preAdapted(getPreADAPT(s, o, r))*

- The access permission decision is defined as:

- $allowed(s, o, r) \Rightarrow preD(s, o, r)$;

  This predicate denotes that the subject s is granted the right r to access the object o, if the adaptation process for this access request is successful.

$$ended(s, o, r) \Rightarrow \left(\begin{array}{c} \neg preD(s, o, r) \\ \wedge \\ \bigvee\limits_{(o', r') \in E} allowed(s, o', r') \end{array} \right)$$

  This predicate denotes that the right r requested by the subject s to access the object o is ended, if the predicate *preD* is false and an alternative request is issued.

  where $E = getPreAltReq(s, o, r)$ and the symbol '$\Rightarrow$' denotes the logical implication.

3.4.6 The CA-UCON$_{onD}$ Model

In the CA-UCON$_{onD}$ model, there is no pre-adaptation; adaptation can only take place during access. If s is subject, o is object and r an access right, we let $onD(s, o, r)$ denote a predicate which is true if the ongoing adaptation is successful and false otherwise. We also denote by $stopped(s, o, r)$ a predicate which is true if the access has been stopped. The CA-UCON$_{onD}$ core model is composed of the following elements:

- S: set of subjects, *ATT(S)*: set of subject attributes, O: set of objects, *ATT (O)*: set of object attributes
- AD: set of adaption strategies (or actions)
- $onCON$: set of ongoing-conditions elements
- T: time domain
- $onAdapted : 2^{onCON} \times AD \times T \rightarrow \{true, false\}$

  onAdapted(c, a, t) is a boolean function that performs action a until all the conditions in c evaluate to *true*, in which case the function returns *true*; otherwise the function returns *false* after t time-units have elapsed since the execution of the action a started.

- $getOnADAPT : S \times O \times R \rightarrow 2^{onCON} \times AD \times T$

  getOnADAPT(s,o, r) returns a tuple *(c, a, t)* where c is the set of all ongoing-conditions required for the subject s to keep the right r upon the object o during access, a is the adaptation action to be performed if any of the ongoing-conditions does not hold, and t is the time-out for this adaptation process.

- $getOnAltReq : S \times O \times R \rightarrow 2^{O \times R}$

  getOnAltReq(s, o, r) denotes the set of alternative requests that can be made on behalf of the subject s when the initial request of the access right r upon the object o fails during access due to environmental conditions.

- *onD(s, o, r) = onAdapted(getOnADAPT(s, o, r))*

- *allowed(s, o, r)* $\Rightarrow$ *true*

- The predicate *stopped* is defined as follows:

$$stopped(s, o, r) \Leftarrow \left(\begin{array}{c} \neg onD(s, o, r) \\ \wedge \\ \bigwedge\limits_{(o', r') \in F} stopped(s, o', r') \end{array} \right)$$

  where $F = getOnAltReq(s, o, r)$ and the formula $V \Leftarrow W$ means that W implies V.

3.5 Expressive Power of the CA-UCON Model

In this section we show that the UCON model can be specified in CA-UCON model and so all the security models that can be specified in UCON, such as Role-Based Access Control (RBAC) and Digital Rights Management (DRM). As mentioned in the previous section, the authorisation and obligation family core models of CA-UCON are identical to those of UCON. Rest to prove that the condition family core models of UCON can be modelled by the adaptation family core models of CA-UCON.

Indeed, the UCON$_{preC}$ model is a special case of CA-UCON$_{preD}$ model where:

- $AD = \{skip\}$, where *skip* is a special action that does nothing and lasts one time-unit.
- $T = \{1\}$, the unique time-out is one time-unit.
- $getPreAltReq(s,\ o,\ r) = \phi$, for all $(s, o, r) \in S \times O \times R$.

Similarly, the UCON_{onC} model is a special case of CA-UCON_{onD} model where:

- $AD = \{skip\}$, where *skip* is a special action that does nothing and lasts one time-unit.
- $T = \{1\}$, the unique time-out is one time-unit.
- $getOnAltReq(s,\ o,\ r) = \phi$, for all $(s, o, r) \in S \times O \times R$.

CA-UCON model can act as UCON model, if the condition of the access request is met where the adaptation in this case is skipped (no adaptation action is performed).

3.6 Summary

In this chapter, we proposed a new model called Context-Aware and Adaptive Usage Control model (CA-UCON) as novel architecture which extends the traditional UCON model to enable adaptation to environmental changes in the aim of preserving continuity of access. The major innovative aspect of this model is the integration of continuity of usage decision and dynamic adaptation to changes in the environmental or system context, so as to ensure continuity of usage. We then propose a novel computational model of our CA-UCON architecture. This model is formally specified as a finite state machine. It demonstrates how the access request of the subject will be handled in CA-UCON model. The formal definition of CA-UCON

model is presented where the extension of the original UCON architecture can be understood from this model.

In the following chapter, the formal specification of CA-UCON model is presented using the mathematical notation of CCA [96] and later used to analyse the behaviour of the model.

Chapter 4

Formal Specification of CA-UCON Model in *CCA*

Objectives:

- Present an overview of mathematical notation of *CCA*
- Give ambient-based model of CA-UCON model
- Present formal specification of CA-UCON model in *CCA*

4.1 Introduction

In this chapter, we represent a formal specification of the CA-UCON model by using the Calculus of Context-aware Ambient (CCA). This mathematical notation is considered suitable for mobile and context-aware systems and has been preferred over alternatives for the following reasons: (i) Mobility and Context awareness are primitive constructs in CCA; (ii) A system's properties can be formally analysed; (iii) Most importantly, CCA specifications are executable allowing early validation of system properties and accelerated development of prototypes. This is based on the concept of an ambient that was first introduced in calculus of mobile ambient [91]. An ambient is an entity that describes a component (e.g., process, location, device, etc.). Using CCA, the model along with all ambient and external and internal interactions can be formalised. Below, we discuss CCA's syntax with regards to processes and capabilities.

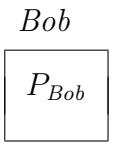
4.2 Overview of *CCA*

To integrate and incorporate computers into general activities, pervasive computing is the new paradigm being used. Pervasive computing allows applications in a distributed system to support mobility and context-awareness. Entities (i.e. devices and users) can be mobile. A context-aware system must be able to process information regarding its current context within its environment. It must then be able to use this information to change the way it behaves and match the current conditions. To model such situations effectively, CCA was proposed by [96]. CCA builds upon a previous calculus known as Mobile Ambients [91] whilst introducing new ideas. It enables ambient (e.g., software, devices, locations, etc.) and processes to have an awareness of the conditions and context in which they are executed. The re-

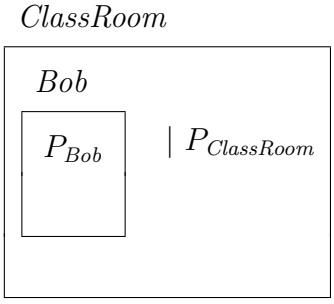
sulting process calculus is flexible and powerful, emphasising context awareness and mobility.

4.2.1 Modelling in CCA

The Calculus of Context-aware Ambients (CCA) is a process calculus. It is based on the idea of ambients (as defined above) and is used to model ambients in terms of capability, process and location. In this section, the modelling of a user and its surrounding environment is shown. From this example, it can be understood how to model ambients in general. Before moving onto the example, let us set out the actors. We specify an arbitrary user Bob whose behaviour is described by the process P_{Bob} as:

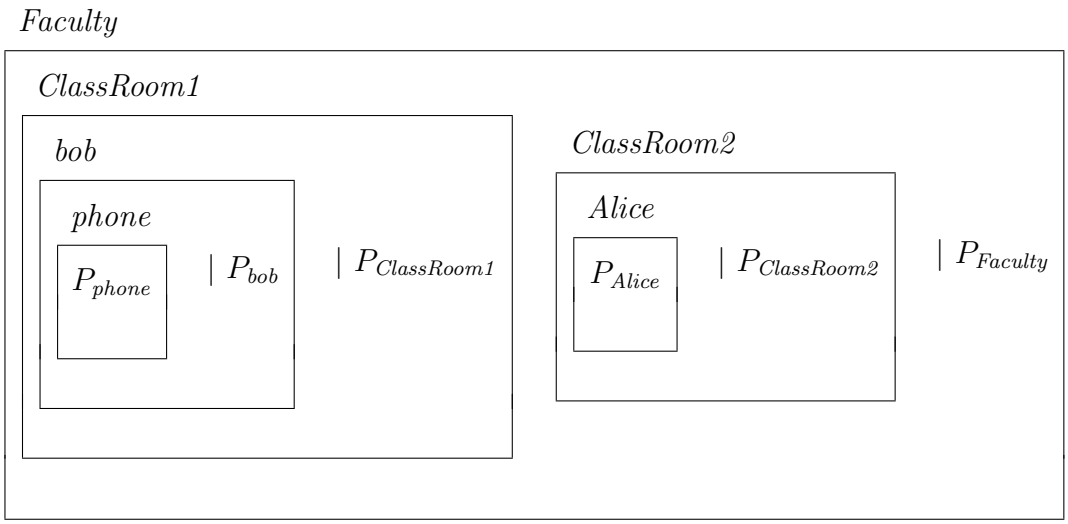


An entity in CCA has a location (similar to the real world). Within CCA the location of an ambient is modelled as a parent ambient. For instant if Bob is in an class room, then the *ClassRoom* ambient will be the parent of ambient Bob and represent his location:



Each ambient has a context which is defined by the ambients around them. For example, Bob's context is defined by the classroom, possible other users, his phone

which he is carrying etc.



So, the user Bob's environment is represented by the variety of the existed ambients which are around it such as in above example, (faculty, classroom1, phone, classroom2 and Alice), which in this situation the user Bob is one of the ambient in this environment. The textual representation of the above graph can be shown as follows:

$$
\begin{aligned}
faculty \quad & [\ classroom1[\ bob[\ phone[P_{phone}] \mid P_{bob}] \mid P_{classroom1}] \\
& \mid classroom2[\ alice[\ P_{alice}] \mid P_{classroom2}] \\
& \mid P_{faculty}\ ]
\end{aligned}
$$

4.2.2 Syntax of CCA

Within this section we present the syntax of CCA notation. In CCA, the most basic entities are names. This is similar to π-calculus [64][91]. Names are given to all ambients whether they are users, devices or locations. We shall define 4 syntactic categories with CCA. These are: processes P, capabilities M, locations α and context expressions κ. Names are always written in lower case letters e.g. n, x

and y etc. A list of names is denoted by $\tilde{y}$ and $|\tilde{y}|$ represents the size of the list.

4.2.2.1 Processes

As can be seen in Table 4.1 the process 0 concludes without actions. If two processes P and Q are running in parallel, this will be denoted by $P|Q$. To limit the scope of a name, the following notation is used: $(\nu n)\ P$, indicates that the scope of n is limited to P. The sign of replication is !, so the replication process $!P$ signifies a process that can create a new replica of the process P whenever needed, i.e. $!P \equiv P|!P$. The process $n[P]$ indicates an ambient n whose behaviour is described by P. Further, the square brackets [and] indicate the boundaries of an ambient's behaviour. Whatever is within the brackets is the ambient's behaviour.

Table 4.1: Syntax of CCA: processes

P, Q	Description
0	inactivity
$P\|Q$	parallel composition
$(\nu n)\ P$	name restriction
$n[P]$	an ambient
$!P$	repetition
$\kappa?M.P$	context-guarded capability
$x \triangleright (\tilde{y}).P$	process abstraction

A context expression denotes the condition that must be matched by the environment of the executing process. The context-guarded prefix $k?M.P$ is a process which delays and does not execute the capability M followed by the process P until the condition (which is the context expression k) fulfils the surrounding environment. The context-guarded prefix is very similar to the $If - Then$ statement in its functionality. The (If condition $Then$ action) statement also performs the action if the condition(s) are satisfied. The use of context-guarded prefix is one of the two main mechanisms for context acquisition in CCA. The second mechanisms of context acquisition (which is discussed below) is the invoking of a process abstraction.

Lastly, the process abstraction $x \triangleright (\tilde{y}).P$ indicates the linking of the name x to the process P, where $\tilde{y}$ is a set of *formal parameters*. The linking is restricted to the executing ambient where in the process abstraction is defined. So a name x can be linked to many processes in a number of distinct ambients. E.g. name x is linked to process P in ambient n and to process Q in ambient m. An invocation to a process abstraction named x is performed by the executing ambient by carrying out the capability $\alpha\ x\langle\tilde{z}\rangle$ where α denotes the position in which the process abstraction is specified and $\tilde{z}$ is the set of *actual parameters*.

4.2.2.2 Location

Since ambients can liaise with each other, the location is an important parameter which is utilised as a reference by the communicating ambients. An ambient can communicate with a parent ambient, child ambient or sibling ambient. So, in table 4.2 the location α can be '$\uparrow$ ' which indicates any parent, '$n \uparrow$' for a definite parent ambient named n, '$\downarrow$' which denotes any child, and '$n \downarrow$' for a definite child ambient named n, the '$::$' which refers to any sibling, and '$n ::$' signifies a specific sibling ambient named n. However, it is possible for an ambient to utilise ϵ (empty string) to refer to itself.

Table 4.2: Syntax of *CCA*: location

α	Description
$\uparrow$	any parent
$n \uparrow$	parent n
$\downarrow$	any child
$n \downarrow$	child n
$::$	any sibling
$n ::$	sibling n
ϵ	local

4.2.2.3 Capabilities

As can be seen in Table 4.3 there are two mobility capabilities, defined in *CCA* [96], which make it possible for an ambient to move in its environment. These are *in* and *out*. An ambient can execute the capability 'in n' to move into a sibling ambient named n, and the capability *out* allows an ambient to move out of its parent ambient.

Table 4.3: Syntax of *CCA*: capabilities

M	Description
`in` n	move into ambient n
`out`	move out
$\alpha\ x\langle\tilde{y}\rangle$	process call
$\alpha\ (\tilde{y})$	input
$\alpha\ \langle\tilde{y}\rangle$	output
`del` n	delete ambient n

A process call $\alpha\ x\langle\tilde{y}\rangle$ executes like the process linked to x at location α, where each *actual parameter* in $\tilde{y}$ replaces each occurrence of the corresponding $formal$ *parameter*. A process call can only occur if the corresponding process abstraction is accessible at the location specified. Ambients are able to send and receive messages. Using the capability $\alpha\ \langle\tilde{y}\rangle$ an ambient is able to send a list of names $\tilde{y}$ to a location α. An ambient can execute the capability $\alpha\ (\tilde{y})$ to receive in the variables in $\tilde{y}$ a set of names from a location α.

Finally, the capability *del* n removes an ambient named n. However, this only occurs if that ambient is of the form $n[0]$, i.e. an empty ambient. Necessarily ambient n has to be located at the same level as that capability. I.e. the ambient executing the *del* capability must be the parent ambient of n. For example, the process `del` $n.P \mid n[0]$ reduces to P. The capability del can be viewed as a garbage collector that removes empty ambients which have no computations left to perform, i.e. $n[0]$.

4.2.3 Context model

In a similar manner to Mobile Ambients MA, the ambient is the basic construction to symbolize any entity in any system in CCA. As mentioned previously, an ambient has a name, a boundary and can host and be nested inside other ambients. This allows a set of ambients to be described hierarchically. A hole, which is denoted by $\odot$, is a special context that symbolizes the position of the executing process in the system. For example, assume a system is represented by the process $P \mid n[Q \mid m[R \mid S]]$, then the context of the process R in that system is $P \mid n[Q \mid m[\odot \mid S]]$. The context of the ambient named m is $P \mid n[Q \mid \odot]$, and that of ambient n is $P \mid \odot$. The grammar in Table 4.4 details the contexts of CCA processes. A property of a context is called context expression (CE in short).

Table 4.4: Syntax of contexts

E	Description
0	nil
$\odot$	hole
$n[E]$	location
$E\|P$	parallel composition
$(\nu n)\ E$	restriction

In Table 4.4, P is any process, E stands for the context and n represents a name. The context 0 represents an empty context, i.e. it does not render any information. The hold $\odot$, denotes the position of a process in that process's context. The context $n[E]$ indicates that the context E is the internal context of the ambient n. The context $P|E$ signifies that the process P executes with the context E in parallel; to rephrase, the context E is an element of the context of process P. The context $(\nu n)\ E$ denotes a name restriction, such that the scope of the name n is limited to the context E.

Table 4.5 presents the algebraic semantics of contexts. The first equality (cont-0) signifies that a context with the value of 0 is a nil context; in other words it would have no affect on other contexts in parallel execution with it. (cont-1) signifies that parallel compositions of context are commutative, while (cont-2) indicates to them being associative. The equality (cont-7) denotes that equality is propagated across scope. (cont-8) states that equality is also propagated across parallel composition and (cont-9) states that equality propagates through ambient nesting.

Table 4.5: Algebraic semantics of contexts

$E \mid 0$	$=$	E	(cont-0)
$E_1 \mid E_2$	$=$	$E_2 \mid E_1$	(cont-1)
$E_1 \mid (E_2 \mid E_3)$	$=$	$(E_1 \mid E_2) \mid E_3$	(cont-2)
$(\nu n)(\nu m)\ E$	$=$	$(\nu m)(\nu n)\ E$	(cont-3)
$(\nu n)\ E_1 \mid E_2$	$=$	$(\nu n)\ (E_1 \mid E_2)$ if $n \notin \mathbf{fn}(E_2)$	(cont-4)
$(\nu n)\ m[E]$	$=$	$m[(\nu n)\ E]$ if $n \neq m$	(cont-5)
$(\nu n)\ 0$	$=$	0	(cont-6)
$E_1 = E_2$	$\Rightarrow$	$(\nu n)\ E_1 = (\nu n)\ E_2$	(cont-7)
$E_1 = E_2$	$\Rightarrow$	$E_1 \mid E_3 = E_2 \mid E_3$	(cont-8)
$E_1 = E_2$	$\Rightarrow$	$n[E_1] = n[E_2]$	(cont-9)

4.2.4 Context Expressions

Table 4.6 details the Context Expressions (CEs in short). The CE $True$ always indicates the value true. The CE $\bullet$ equates to true for the hole context only. To explain, it represents the process evaluating that context expression. For the CE $n = m$, it applies only if the names n and m are lexically identical, i.e. they match. Such a CE is vital for examining whether the contents of two messages are identical.

The first order operators $\neg$ (negation), $\wedge$ (and) and $\exists$ (there exist) develop their standard meaning to CEs.

A CE $\kappa_1|\kappa_2$ is valid for a context if that context is a parallel composition of two contexts. These must be such that κ_1 holds for one and κ_2 holds for the other. A CE $n[\kappa]$ holds for a context if that context is an ambient named n such that κ holds inside that ambient. A CE $new\ n\kappa$ holds for a context if that context restricts the name n to another context for which κ holds. A CE $\oplus\kappa$ holds for a context on the condition that the context has a child context for which κ holds. For a CE $\text{✧}\kappa$ to hold for a context there must be a sub-context present, somewhere in that context, for which κ holds. The operator ✧ is known as *somewhere modality* while $\oplus$ is known as *spatial next modality*.

Table 4.6: Syntax of CCA: context expressions

κ	Description
$True$	true
$\bullet$	hole
$n = m$	name match
$\neg\kappa$	negation
$\kappa_1\|\kappa_2$	parallel composition
$\kappa_1 \wedge \kappa_2$	conjunction
$newn\kappa$	revelation
$\oplus\kappa$	spatial next modality
$\text{✧}\kappa$	somewhere modality
$\exists x.\kappa$	existential quantification

It is possible logically to infer some derivates. For example, $False\ = \neg True$, where both sides implies the value $false$. The disjunction relation $\kappa_1\ \vee\ \kappa_2$ is logically equivalent to $\neg(\neg\kappa_1\ \wedge\ \neg\kappa_2)$. The implication of $\kappa_1\ \Rightarrow\ \kappa_2$ can be expressed as $\neg\kappa_1\ \vee\ \kappa_2$, with both having the same logical value. Lastly, a logical equivalence of two contexts $\kappa_1\ \Leftrightarrow\ \kappa_2$ is equivalent to $(\kappa_1\ \Rightarrow\ \kappa_2)\ \wedge\ (\kappa_2\ \Rightarrow\ \kappa_1)$.

4.3 Ambient-based model for CA-UCON model

As depicted in the previous chapter, the CA-UCON model is represented in a finite state machine in order to show how the requests are handled. Accordingly, the CA-UCON model consists of the following states: *initial* state, *requesting* state, *accessing* state, *preadaptation* state, and *onadaptation* state. The behaviour of the user that initiates an access request is also represented by an ambient named *subject*. In addition, specific ambients are used to check the authorisation, obligation and condition requirements. However, in this section, we shall now propose an ambient-based model for CA-UCON, where each state within CA-UCON is modelled as an ambient. Below in Figure 4.1, an ambient-based CA-UCON model is provided. Following this, a description shall be given regarding how to change this graphical model into its textual equivalent in *CCA*.

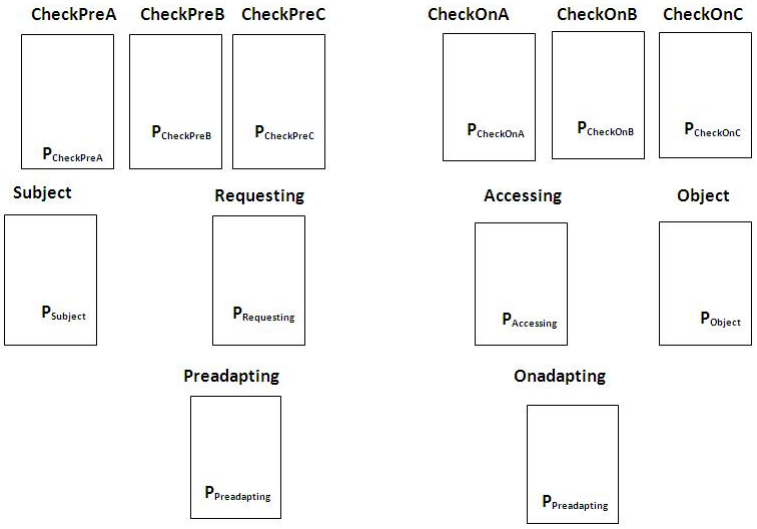


Figure 4.1: Ambient-based Model for CA-UCON Model

We begin our CA-UCON model by representing each entity as an ambient. So, if the subject submits a request to access an object, the request goes to the *requesting* ambient, which checks for *preauthorization*, *preobligation* and *precondition*. If *preauthorization* or *preobligation* is not met, then the request is denied. However, if

they are met but the *precondition* is not met, the system adapts to the new situation based on the context. This happens in *accessing* ambient during the access as well. The above figure is represented textually through the following *CCA* process:

$$
\begin{aligned}
&subject[P_{subject}] \mid requesting[P_{requesting}] \mid accessing[P_{accssing}] \\
&preadapting[P_{preadapting}] \mid onadapting[P_{onadapting}] \\
&\mid preauthorisation[P_{preauthorisation}] \mid preobligation[P_{preobligation}] \\
&\mid precondition[P_{precondition}] \mid onauthorisation[P_{onauthorisation}] \\
&\mid onobligation[P_{onobligation}] \mid oncondition[P_{oncondition}]
\end{aligned}
$$

Where P_x in this textual model is the process modelling the behaviour of the related ambient x. These processes are defined in details in the following section.

4.4 Formalising the CA-UCON Model in *CCA*

Here, we represent the process of formalisation for the CA-UCON model. This formalisation is based on the functional aspects that control access in order to *permit* or *deny* a service. Therefore, we formalise all the ambients in the CA-UCON model as well as the interactions among them both before and during the access request [**?**]. The CA-UCON formalisation process is illustrated utilizing the mathematical notation of *CCA*.

4.4.1 Notation

In the following formalisation process of the CA-UCON model, we use a different format of letters in order to distinguish between constants and names (variables). The names written in lowercase letters are known as variables, while the names written in uppercase letters are known as constants. In Table 4.7 below, the list of

Table 4.7: Constants

Constants	
Notation	Description
PERMIT	permit the access
DENYA	deny Authorisation
DENYB	deny Obligation
DENYC	deny Condition
REVOKEA	revoke Authorisation
REVOKEB	revoke Obligation
REVOKEC	revoke Condition
END_USAGE	terminate usage
ENDED_SUCCESSFULLY	usage ended successfully

Table 4.8: Variables

Variables		
Notation	Description	values
s	a user id	201,202, 203
o	object id	lect1, tut3,test4
preA	preauthorization	0,1
preB	preobligation	0,1
preC	precondition	0,1
preD	preadaptation	0,1
onA	onauthorisation	0,1
onB	onobligation	0,1
onC	oncondition	0,1
onD	onadaptation	0,1

variables is shown and in the subsequent table 4.8, the list of constants is shown.

4.4.2 *Subject* Ambient

This ambient is responsible for submitting the access request to the *requesting* ambient; it then waits for the reply to this access request, whether *permit* or *deny* . The user eventually ends the usage by sending the message END_USAGE to the *accessing* ambient. This behaviour is modelled as follows:

$$
\begin{aligned}
P_{subject} \mathrel{\widehat{=}} !&requesting :: \langle s, o, r\rangle.requesting :: (reply, o, r).\{ \\
&(reply = PERMIT)?accessing :: (x, o, r).0 \\
&\mid accessing :: \langle \text{END\_USAGE}, o, r\rangle.0 \\
\}&
\end{aligned}
$$

Where s is the user id, o the object id and r the access right requested.

4.4.3 *Requesting* Ambient

This ambient handles all the access requests sent by the subject in order to access an object. It receives an access request from the subject ambient and checks the *preauthorisation*, *preobligation* and *precondition* by sending the access request to the *checkPreA*, *checkPreB* and *checkPreC* ambients, respectively. If the *preA* and *preB* and *preC* are true (i.e. 1), the subject is permitted to access the requested object. But if *preA* (or *preB*) is not true (i.e. 0), the reply *DENYA* (or *DENYB* respectively) is sent to the subject.

However, if *preA* and *preB* are true but *preC* is false, the request is sent to the *checkPreD* ambient in order to adapt to the new situation if possible. The reply *DENYC* is sent to the subject if the adaptation fails (i.e. $preD = 0$). This behaviour is modelled as follows:

$$
\begin{array}{l}
P_{requesing} \mathrel{\widehat{=}} \; !::(s,o,r).checkPreA::\langle s,o,r\rangle.checkPreA::(preA). \\
\left\{\begin{array}{l}
(preA=1)?checkpPreB::\langle s,o,r\rangle.checkPreB::(preB). \\
\left\{\begin{array}{l}
(preB=1)?checkPreC::\langle s,o,r\rangle.checkPreC::(preC). \\
\left\{\begin{array}{l}
(preC=1)?subject::\langle PERMIT,o,r\rangle. \\
preUpdate::\langle s,o,r\rangle.preUpdate::().accessing::\langle s,o,r\rangle.0 \\
|(preC=0)?preadapting::\langle s,o,r\rangle.preadapting::(preD). \\
\left\{\begin{array}{l}
(preD=1)?subject::\langle PERMIT,o,r\rangle. \\
preUpdate::\langle s,o,r\rangle.preUpdate::().accessing::\langle s,o,r\rangle.0 \\
|(preD=0)?subject::\langle DENYC,o,r\rangle.0
\end{array}\right.
\end{array}\right. \\
|(preB=0)?subject::\langle DENYB,o,r\rangle.0
\end{array}\right. \\
|(preA=0)?subject::\langle DENYA,o,r\rangle.0
\end{array}\right.
\end{array}
$$

4.4.4 *Accessing* Ambient

This ambient is responsible for continuously controlling all the permitted access requests that it receives from the *requesting* ambient. This is achieved through sending the access request to the *checkOnA*, *checkOnB* and *checkOnC* ambient, and receiving the reply in the following variables: *onA*, *onB* and *onC*. If *onA* or *onB* is false, the reply *REVOKEA* or *REVOKEB* is sent to the subject, respectively.

However, if *onA* and *onB* are true but *onC* is false, the access request is sent to *checkOnD*, whereupon it receives the parameter *onD*; if this is true, the subject will be permitted to continue to access the object until the user ends the usage. Otherwise, the reply *REVOKEC* is sent to the subject ambient and the usage terminates immediately. This behaviour is modelled as follows:

$$
\begin{aligned}
&P_{accessing} \mathrel{\widehat{=}} \ !::(s,o,r).checkOnA::\langle s,o,r\rangle.checkOnA::(onA).\\
&\left\{\begin{array}{l}
(onA=1)?checkOnB::\langle s,o,r\rangle.checkOnB::(onB).\\
\left\{\begin{array}{l}
(onB=1)?checkOnC::\langle s,o,r\rangle.checkOnC::(onC).\\
\left\{\begin{array}{l}
(onC=1)?OnUpdate::\langle s,o,r\rangle.OnUpdate::().accessing::\langle s,o,r\rangle.0\\
|(onC=0)?Onadapting::\langle s,o,r\rangle.Onadapting::(onD).\\
\left\{\begin{array}{l}
(onD=1)?OnUpdate::\langle s,o,r\rangle.OnUpdate::().accessing::\langle s,o,r\rangle.0\\
|(onD=0)?subject::\langle REVOKEC,o,r\rangle.0
\end{array}\right.
\end{array}\right.\\
|(onB=0)?subject::\langle REVOKEB,o,r\rangle.0
\end{array}\right.\\
|(onA=0)?subject::\langle REVOKEA,o,r\rangle.0
\end{array}\right.\\
&|subject::(X).().PostUpdate::\langle s,o,r\rangle.postUpdate::(X).\\
&subject::\langle ENDED\_SUCCESSFULLY,o,r\rangle.0
\end{aligned}
$$

4.4.5 *Preadapting* Ambient

When the *precondition* does not hold, the *requesting* ambient forwards the access request to this ambient which then attempts to adapt to the context by performing specific actions. In addition, it is able to issue an alternative request which is appropriate to the current situation. Otherwise, the access request is simply *denied*, when no adaptation is possible. This behaviour is modelled as follows:

$$P_{preadapting} \mathrel{\widehat{=}} \ !requesting::(s,o,r).P.requesting::\langle preD\rangle.0$$

where P stands for the pre-adaptation actions or alternative request, which must be done by the *preadapting* ambient. The decision is calculated in the variable *preD* and sent to the *requesting* ambient. The actual specification of the process P is application dependent.

4.4.6 *Onadapting* Ambient

When the *oncondition* is false during the access, the *accessing* ambient forwards the access request to the *onadapting* ambient which attempts to adapt to the new situation in order to maintain continuity of usage of the resource. Like the *requesting* ambient, it is capable of issuing an alternative request depending on the context. If no adaptation is possible, the access is simply revoked. This behaviour is modelled as follows:

$$P_{onadapting} \mathrel{\widehat{=}} accessing :: (s, o, r).P.accessing :: \langle onD \rangle.0$$

where P stands for on-adaptation actions or alternative request, which must be performed by the *onadapting* ambient. The decision is calculated in the variable *onD* and sent to the *accessing* ambient.

4.4.7 *CheckPreA* Ambient

This ambient receives an access request from the *requesting* ambient and checks whether the *preauthorisation* requirements are met by the subject. It then sends the decision *preA* to the *requesting* ambient. This behaviour is achieved as follows:

$$P_{checkPreA} \mathrel{\widehat{=}} !requesting :: (s, o, r).P.requesting :: \langle preA \rangle.0$$

where the process P models the *preauthorisation* requirements, which again are application dependent.

4.4.8 *CheckPreB* Ambient

This ambient receives an access request from the *requesting* ambient and checks whether the *preobligation* requirements are met by the subject. Then, it sends the decision *preB* to the *requesting* ambient. This behaviour is achieved as follows:

$$P_{checkPreB} \widehat{=}\ !requesting :: (s, o, r).P.requesting :: \langle preB \rangle.0$$

where the process P models the *preobligation* requirements.

4.4.9 *CheckPreC* Ambient

This ambient receives an access request from the *requesting* ambient and checks whether the *precondition* requirements are met by the environment. Then, it sends the decision *preC* to the *requesting* ambient. This behaviour is achieved as follows:

$$P_{checkPreC} \widehat{=}\ !requesting :: (s, o, r).P.requesting :: \langle preC \rangle.0$$

where the process P represents the precondition requirements.

4.4.10 *CheckOnA* Ambient

This ambient plays similar role for *onauthorisation* requirements like the *checkPreA* ambient for *preauthorisation* requirements. The decision whether the *onauthorisation* requirements are met is returned to the *accessing* ambient via the variable *onA*. This behaviour is specified as follows:

$$P_{checkOnA} \mathrel{\widehat{=}} !accessing :: (s, o, r).P.accessing :: \langle onA \rangle.0$$

where the process P models the onauthorisation requirements.

4.4.11 *CheckOnB* Ambient

Similarly to the *checkOnA* ambient, this ambient plays the same role for *onobligation* requirements as the *checkPreB* ambient for *preobligation* requirements. The decision whether the *onobligation* requirements are met is returned to the *accessing* ambient via the variable *onB*. This behaviour is specified as follows:

$$P_{checkOnB} \mathrel{\widehat{=}} !accessing :: (s, o, r).P.accessing :: \langle onB \rangle.0$$

where P represents the *onobligation* requirements.

4.4.12 *CheckOnC* Ambient

As for this ambient, it receives an access request from the *accessing* ambient and checks whether the *oncondition* requirements are met by the environment. The decision is sent to the *accessing* ambient in the variable *onC*. This behaviour is specified as follows:

$$P_{checkOnC} \mathrel{\widehat{=}} !accessing :: (s, o, r).P.accessing :: \langle onC \rangle.0$$

where the process P models the *oncondition* requirements.

4.5 Summary

This chapter, we introduced the concept of an ambient-based model and described its characteristics. We then presented the mathematical notation of *CCA*, including its syntax, with regard to the capabilities and processes. *CCA* is an appropriate mathematical notation for modelling mobile applications that are context-aware. In addition, we presented the graphical model of CA-UCON model based on ambient concept. Finally, we demonstrated the formalisation of CA-UCON model with all external and internal interactions between its components using the *CCA* notation.

In chapter 5, we will demonstrate the U-learning system as a real world case study in order to evaluate our CA-UCON model. We then use the execution environment of CCA to validate the properties of CA-UCON model.

Chapter 5

Case Study

Objectives:

- Present an overview of u-learning system and its infrastructure.
- Present the modelling of u-learning system in CA-UCON.
- Give a CCA specification of the u-learning system.
- Use ccaPL to validate some properties of CA-UCON model.

5.1 Introduction

In this chapter, we evaluate our CA-UCON model with a case study in the ubiquitous learning system (u-learning) in order to show how our model functions in this environment. Firstly, we give an overview of u-learning systems and the technologies and infrastructures that have been used in such systems. Secondly, we propose a CA-UCON model for the u-learning system and specify all the policies. Thirdly, we provide the formal specifications for this model in *CCA*; these specifications will be analysed using the execution environment of *CCA*, which will demonstrate how the properties of the CA-UCON model can be validated.

5.2 Ubiquitous Learning (U-Learning)

In this section we provide an overview of u-learning (derived from the literature) and describe the technologies and infrastructures used in u-learning systems.

5.2.1 Overview

The notion of ubiquitous learning (u-learning) has become more prevalent since the development of new ICT technologies facilitating ubiquitous computing, and since electronic communications have become so widespread. However, the reason behind the recent evolution of u-learning is that there are now increasing demands for different learning methods to solve the challenges of learning in a flexible manner[32]. U-learning is defined as the ability of a ubiquitous computing system to sense the learner's situation and to offer him/her adaptive contents based on his/her context (this is called context-awareness). U-learning is described as a special case of m-learning, with the notion of learning being based on context. Thus, the aim of u-learning is that it considers the context of the learner in order to provide

her/him with learning contents at appropriate times and in appropriate situations. U-learning is similar to m-learning in terms of mobility but u-learning goes beyond m-learning by including a high level of embeddedness. This is because u-learning is able to determine the learner's context using a number of sensors in order to detect and gather information pertaining to the learner and his/her particular environment [59][54].

5.2.2 U-learning Technologies and Infrastructure

U-learning is based on an extremely large range of wireless, mobile, wearable, portable and embedded devices. Figure 5.1 illustrates that u-learning infrastructural technology has two significant factors: devices that process the information, and the communication network. In addition, ubiquitous information processing is incorporated into office and home appliances as well as handheld and mobile devices; also, they are able to communicate with each other directly [114].

With regards to communication networks, the main distinction is between heterogeneity and ad hoc formations. The former indicates that the network is a mixture of complementary technologies, not just a single technology. This kind of heterogeneity is not observable for the learners that use the learning services. In this case, the network should have a high level of synthetic intelligence in order for it to select the appropriate technology and the best operational approach for each user-initiated application. The latter denotes that the network is created based on the needs and circumstances of the learner, and is not predefined. The idea of ad hoc networks is that they are applied on network topology with the dynamism constantly changing [107].

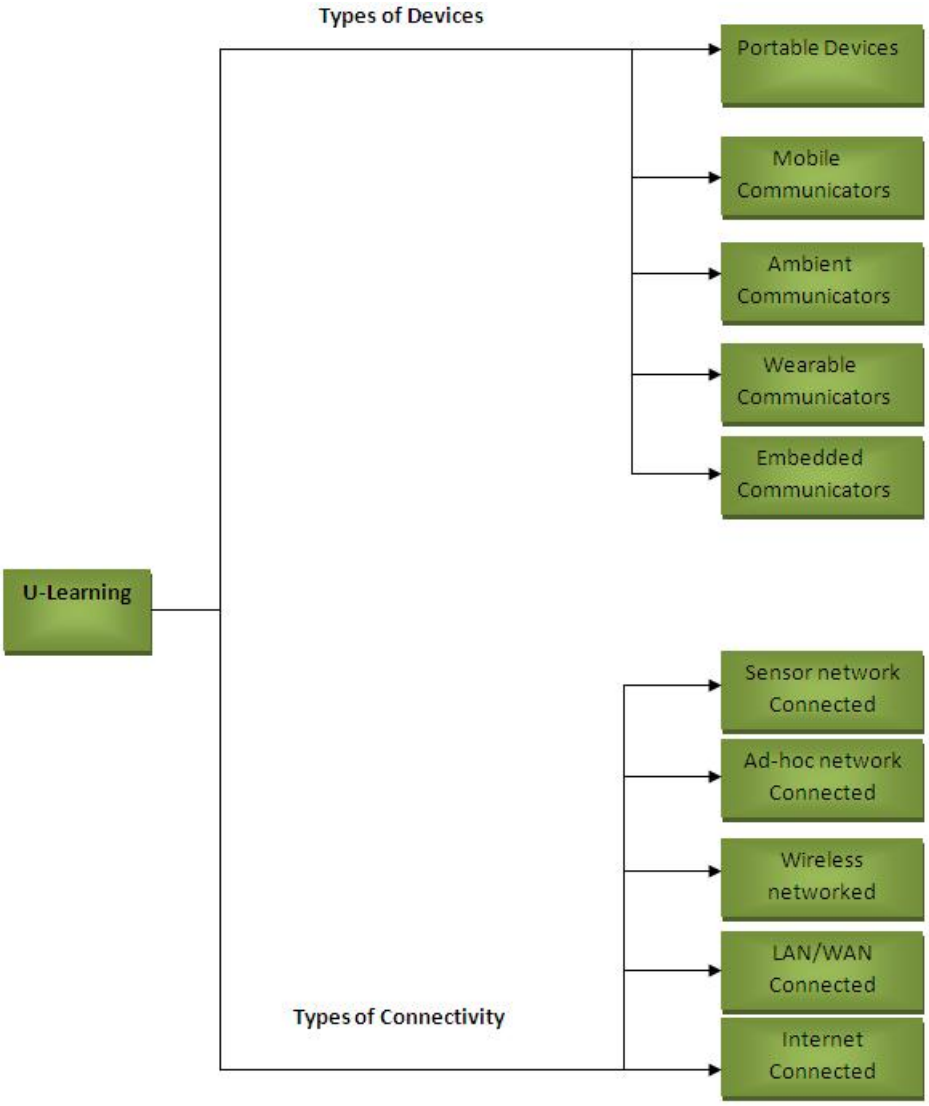


Figure 5.1: U-learning types of devices and connectivity [114]

5.3 Modelling of a U-learning System in CA-UCON

5.3.1 U-learning Services

A u-learning system considers the context of learners, devices, services and environment in order to provide services such as u-lectures, u-tutorials and u-tests. Therefore, a u-learning system allows the learners to use their portable devices, such as smart phones, laptops and PDAs, to connect wirelessly to various wireless networks, such as Wi-Fi spots, WLANs and 3G terminals, in order to access u-learning services anytime, anywhere. Thus, the u-learning system delivers the content of services to the learner as appropriate and adaptable content, based on the current context. A u-learning system offers the following services:

1. U-Lecture: this kind of service offers learners all possible lecture materials, which they can request and access through their mobile devices. There are three different types of u-lecture formats provided by the u-learning system: text, video and audio. In the u-learning system, each format requires a particular context in order for it to be delivered to the learner.

2. U-Test: this service is a formal approach for evaluating the learner's understanding of the u-learning content that has been requested and provided. It assesses the learner's knowledge and based on the results, the learners receive feedback to assist them in further u-learning. The learners utilize their mobile devices in accessing this service.

3. U-Tutorial: this type of service is used by learners to increase their knowledge in a particular subject; it is a useful service, as it provides an opportunity for learners to obtain some form of self-assessment and to receive direct and per-

sonal feedback. This service is a combination of u-lecture and u-test. There are three different types of u-tutorial formats provided by the u-learning system: text, video and audio.

5.3.2 Requirements of the u-learning system

A number of different requirements in the u-learning system must be fulfilled for the learner to gain access. The u-learning system requirements are: authorization requirements, obligation requirements, condition requirements and adaptation requirements.

5.3.2.1 Authorisation Requirements

The authorisation requirements pertaining to the use of the u-learning system must be satisfied before the learner is permitted to utilize it, i.e. the learner must be enrolled. Therefore, we suppose that there are no specific authorisation requirements in order for the learner to access a service.

5.3.2.2 Obligation requirements

In this section we present the obligation requirements for the u-learning system; these must be fulfilled before and during the access request. Accordingly, the obligation requirements in the u-learning system are detailed as follows:

B1: The user must register on the module that includes the requested service before gaining access.
B2: The user must open the module announcement which is related to the requested service for a duration of 10 seconds.

5.3.2.3 Condition requirements

Condition requirements within the u-learning paradigm are defined as context requirements; these are considered by the system before and during the access request in order to permit access and delivery of the requested service to the learner. We present several condition requirements, as follows:

C1: The u-learning system considers the context of user in order to deliver the u-service. So, the only u-service format is allowed for a learner in *driving* context is the audio format.

C2: In the u-learning system at a university, the context of the learner is classified as private or public. The public context is defined as learners being in a location where making a noise is not permitted, such as in a library, seminar or lecture hall. On the other hand, the private context denotes that the learners are in a location where making a noise is permitted, such as in a cafe or at home. Therefore, the system provides u-services in any of the different formats (video, audio or text) but it considers the learner context before and during the access request. So, the only u-service format allowed for a learner in a public context is the text format. However, the system permits access to all the different u-service formats when the learner context is detected as private.

C3: As above, the u-learning system at a university provides u-services in dif-

ferent formats (video, audio and text); however, it considers the availability of the memory of the user's device before and during the access request. Each u-service format requires a different amount of space for memory space in order to facilitate access. If the learner requests a u-service in the video format, audio format or text format the available memory of the device (before the access request) must be more than 5MB, 2MB or 1MB respectively for access to be permitted. However, the available memory of the device (during the access request) must be greater or equal to 1MB for all formats.

C4: In a u-learning system, the battery capacity and bandwidth of the network are considered when the user requests a u-service in the video format. Therefore, if the user's device has a low bandwidth (such as 2G), the battery capacity must be more than 10% (before the access request) in order for access to the video format service to commence. However, during the access request the battery capacity must be more than 5%.

C5: The u-learning system considers the battery capacity before and during the access request of a u-test. Thus, the u-test service has two forms, which are defined as *part of test* and *full test*. If the user requests a *full test*, the battery capacity must be more than 10% in order to access this service. However, the battery capacity must be more than 5% in order to access *part of test*

5.3.2.4 Adaptation requirements

Adaptation requirements are defined as set of adaptation actions that are used by the system to adapt when conditions do not hold. In the u-learning system, there are several conditions, which have been explained in the previous section, and these conditions must be fulfilled before and during the access request. Otherwise, specific actions are performed by the system in order to adapt to a new situation. In addition, the system might issue an implicit request to access a specified alternative object, when there are no adaptation actions available or the adaptation action fails. Several adaptations are explained in the following paragraphs:

D1: If the context of the user is *driving* and the requested service is in video or text formats, the adaptation action is that the system issues an alternative request in order to deliver audio format instead.

D2: If the user requests the service in video or audio format when s/he is in a public place, the adaptation action is such that the system issues an alternative request that enables the user to access the service in text format instead.

D3: Again, if the user requests a u-service in video, audio or text formats, the required minimum available spaces in the memory are 5MB, 2MB and 1MB, respectively. The adaptation action in this case, if the conditions do not hold, is that the system activates garbage collector software in an attempt to make more space available in the memory.

Garbage Collector (GC) is a type of automatic memory management that attempts to dispose of objects or data that are no longer required, releasing the space

that they had engaged.

D4: if the user requests video or audio format when s/he is using low bandwidth and battery capacity is less than 10%, the adaptation action (in this case) is such that the system switches to an available high-bandwidth network.

D5: if the user requests a u-test in 'full test' format when the battery capacity is less than 10%, the adaptation action is such that the system issues an alternative request that delivers a 'part of test' format to the user instead.

5.3.3 Formalisation in CA-UCON

In this section, we present the formalisation of u-learning services based on the CA-UCON model. We adopt a finite set S of subjects and a finite set O of objects. The subjects are learners who are studying at a university and the objects are lecture, tutorial and test.

5.3.3.1 Right

These are the set of rights that the learner must hold in order to access u-service: *download, submit and write*. The *download* right means that the learner is allowed to download a u-learning service (such as a u-lecture, u-tutorial or u-test) if all requirements pertaining to this right have been fulfilled. The *submit* right is used by the learner to submit a u-test. The *write* right means that the learner is allowed to sit a test if the access right has been satisfied.

$$\mathbf{R} = \{\text{download, submit, write}\}$$

5.3.3.2 Authorization

This is the authorization predicate, which checks the attributes of the subject and object in terms of the access right, both before the access and during the access. In this u-learning system, there are no authorization requirements for accessing the u-services. So, the authorization predicates are taken to be true. The authorization predicates are specified as follows:

- $PreA(ATT(s), ATT(o), r) = true.$
- $OnA(ATT(s), ATT(o), r) = true.$

5.3.3.3 Obligation

Obligation consists of three elements *OBS*, *OBO* and *OB*. The *OBS* is the set of subjects who perform the obligation action, while *OBO* is the set of objects on which the obligation is to be performed. The *OB* is the set of obligation actions that the subject must perform in order to access the object. Therefore, the obligation is specified as follows:

- $OBS = S.$
- $OBO = O.$
- $OB = \{register, OpenAnnouncement\}.$

  There are two obligation actions as shown above; the first one is *register*, which means that the learner must register with the requested service. The second obligation action is *OpenAnnouncement*, which means that when the learner requests the u-service, s/he must open the module announcement during the access in order to continue usage of the service.

- $PreOBL \subseteq OBS \times OBO \times OB$.
- $getPreOBL(s, o, r) = \{(s, o, register)\}$.

  This function means that for each user who would like to access any service by using any right, s/he has to register to the module which includes the requested service.

- $OnOBL \subseteq OBS \times OBO \times OB \times T$

- $getOnOBL(s, o, r) = \{(s, o, OpenAnnouncement, 10)\}$.
  This function means that each user has to open the module announcement for at least 10 seconds in order to continue the usage of the service.

- $PreB(s, o, r) \Rightarrow PreFulfilled(getPreOBL(s, o, r))$.

- $allowed(s, o, r) \Rightarrow PreB(s, o, r)$.
- $OnB(s, o, r) \Rightarrow OnFulfilled(getOnOBL(s, o, r))$.

- $stopped(s, o, r) \Leftarrow \neg onB(s, o, r)$.

5.3.3.4 Condition

We use a variable *StudentContext* to denote the context of learner; its possible values are: *driving*, when the context of the learner is *driving*; *public*, when the context of the learner is in a *public* place and *private*, when the context of the learner is in *private* place. Accordingly, the conditions are specified as follows:

C1: The user should not be driving when requesting a u-video or u-text service. This is specified by the condition:

$$(StudentContext \neq driving) \tag{5.1}$$

C2: To access a u-video or u-audio, the user must be in a private place. This is specified by the condition:

$$(StudentContext = Private) \tag{5.2}$$

C3: The user must have sufficient memory available on her/his mobile device, which must be more than 5MB if s/he requests the service in video format, more than 2MB if in audio, and more than 1MB if the requested service is in text format before the access request. However, during the access the memory available must be greater or equal to 1MB for all formats. We use the variable *AvailableMemory* to denote the amount of free space available in the memory. This is specified by the conditions:

$$(AvailableMemory > 5MB) \tag{5.3}$$

$$(AvailableMemory > 2MB) \tag{5.4}$$

$$(AvailableMemory > 1MB) \tag{5.5}$$

$$(AvailableMemory >= 1MB) \tag{5.6}$$

C4: The user may not access the service in video format, if s/he is using a low bandwidth network and the battery capacity is less than 10% before and during the access request. We use the variable *Bandwidth* to denote the signal bandwidth being used for the communication, while we use the variable *BatteryCapacity* to denote how much power the device has.
This is specified by the condition:

$$(Bandwidth = high \vee BatteryCapacity > 10\%) \tag{5.7}$$

C5: The user may not access a full u-test if the battery capacity is less than 10%.
This is specified by the condition:

$$(BatteryCapacity > 10\%) \tag{5.8}$$

PreCon={Eq.(5.1),Eq.(5.2),Eq.(5.3),Eq.(5.4),Eq.(5.5),Eq.(5.7),Eq.(5.8)}

OnCon={Eq.(5.1),Eq.(5.2),Eq.(5.6),Eq.(5.7),Eq.(5.8)}

5.3.3.5 Adaptation

Adaptation requirements are a set of adaptation actions that are used by the system to adapt when specific conditions do not hold. In particular, there are four adaptation actions: *AdaptToVideo*, *AdaptToAudio*, *AdaptToText* and *skip*. We use the variable *gc* to model the garbage collector; this variable can only have two values: 1 (to mean that the garbage collector is activated) or 0 (to deactivate it). The action *skip* does nothing and lasts for only one time-unit. The adaptation actions are specified as follows:

$\boldsymbol{AD}$= {*AdaptToVideo*, *AdaptToAudio*, *AdaptToText*, *Skip*}.

We partition the set O of objects as follows:

$\boldsymbol{O} = O_{video} \bigcup O_{audio} \bigcup O_{text}$.

where O_{video} is the set of video versions of u-lecture, u-tutorial and u-test; O_{audio} is the set of audio versions of u-lecture, u-tutorial and u-test; and O_{text} is the set of text versions of u-lecture, u-tutorial and u-test.

getPreADAPT(s, o, r) returns a tuple (c, a, t) where c is the set of all pre-conditions required to grant the subject s the access right r upon the object o, a is the adaptation action to be performed if any of the pre-conditions do not hold, and t is the time-out for this adaptation process.

getOnADAPT(s,o, r) returns a tuple *(c, a, t)* where c is the set of all ongoing-conditions required for the subject s to keep the right r upon the object o during access, a is the adaptation action to be performed if any of the ongoing-conditions does not hold, and t is the time-out for this adaptation process.

$$getPreADAPT(s, o, download) = getOnADAPT(s, o, download) = (\{Eq.(5.1), Eq.(5.2), Eq.(5.3), Eq.(5.7)\}, AdaptToVideo, 5) \tag{5.9}$$

for $o \in O_{video}$

This function means that if the user requests the video format and if one (or all) of these Eq.(5.1),Eq.(5.2), Eq.(5.3), Eq.(5.7) conditions do not hold, the system will adapt by performing the function $AdaptToVideo$; for a duration of 5 seconds. We suppose that the time taken for the $AdaptToVideo$ action to be performed in this system is 5 seconds.

The action $AdaptToVideo$ is defined as follows:

AdaptToVideo $= (if \neg(Eq.(5.1)) \quad then \quad Skip$

|| $if \neg(Eq.(5.2))$ $then$ $Skip$

|| $if \neg(Eq.(5.3))$ $then$ $gc := 1$

|| $if \neg(Eq.(5.7))$ $then$ $Bandwidth := high$

)

The function *AdaptToVideo* performs the action *Skip*, if the context of the user is *driving* or *public*. However, the action *gc:=1* is performed if the *AvailableMemory* less or equal to 5 MB which means that turn on the Garbage collector. The action *Bandwidth:= high* is performed if the bandwidth is low.

$$getPreADAPT(s, o, download) = getOnADAPT(s, o, download) = (\{Eq.(5.2), Eq.(5.4)\}, AdaptToAudio, 4) \tag{5.10}$$

for $o \in O_{audio}$

This function means that if the user requests the audio format and if one (or all) of these Eq.(5.2), Eq.(5.4) conditions do not hold, the system will adapt by performing the function *AdaptToAudio*; for a duration *4* seconds. We suppose that the time taken for the *AdaptToAudio* action to be performed in this system is *4* seconds.

The action *AdaptToAudio* is defined as follows:

AdaptToAudio = ($if \neg(Eq.(5.2))$ $then$ $Skip$

|| $if \neg(Eq.(5.4))$ $then$ $gc := 1$

)

The function *AdaptToAudio* performs the action *Skip*, if the context of the user

is *public*. However, the action *gc:=1* is performed if the *AvailableMemory* less or equal to 2 MB which means that turn on the Garbage collector.

$$getPreADAPT(s, o, download) = getOnADAPT(s, o, download) = (\{Eq.(5.1), Eq.(5.5)\}, AdaptToText, 3) \tag{5.11}$$

for $o \in O_{text}$

This function means that if the user requests the text format and if one (or all) of these Eq.(5.1), Eq.(5.5) conditions do not hold, the system will adapt by performing the function *AdaptToText*; for a duration of 3 seconds. We suppose that the time taken for the *AdaptToText* action to be performed in this system is *3* seconds.

The action *AdaptToText* is defined as follows:

AdaptToText = $(if \neg(Eq.(5.1)) \ \ then \ \ Skip$
|| $if \neg(Eq.(5.5)) \ \ then \ \ gc := 1$
)

The function *AdaptToText* performs the action *Skip*, if the context of the user is *driving*. However, the action *gc:=1* is performed if the *AvailableMemory* less or equal to 1 MB which means that turn on the Garbage collector.

$$getPreADAPT(s, o, download) = getOnADAPT(s, o, download) = (\{Eq.(5.8)\}, skip, 1) \tag{5.12}$$

for $O \in O_{test}$

This function means that if the user requests a u-test service and if the Eq.(5.8) condition does not hold, which is in this case would be that the battery capacity is less than 10%, the system will perform the *skip* action for a duration *1* second, because the system is unable to make this condition true. However, an alternative request will be issued by system as detailed below.

getPreAltReq(s, o, r) this denotes the set of alternative requests that can be made on behalf of the subject s when the initial request of the access right r upon the object o fails before access due to environmental conditions.

getOnAltReq(s, o, r) denotes the set of alternative requests that can be made on behalf of the subject s when the initial request of the access right r upon the object o fails during access due to environmental conditions.

For the sake of simplicity, we will use the following functions:

$Video : O \longrightarrow O_{video}$; video(o)= the video version of object(o).

$Audio : O \longrightarrow O_{audio}$; audio(o)= the audio version of object(o).

$Text : O \longrightarrow O_{text}$; text(o)= the text version of object(o).

$PartOfTest : O \longrightarrow O_{partoftest}$; PartOfTest(o)= the part of the test version of object(o).

$$getPreAltReq(s, o, download) = getOnAltReq(s, o, download) = \{(Audio(o), download)\} \tag{5.13}$$

for $o \in O_{video} \bigcup O_{text}$

In the functions above, if the user requests a u-service the video or text format when driving, in this case the condition does not hold; the system then issues an alternative request, which delivers that u-service in the audio format instead.

$$getPreAltReq(s, o, download) = getOnAltReq(s, o, download) = \{(Text(o), download)\} \tag{5.14}$$

for $o \in O_{video} \bigcup O_{audio}$

The functions above mean that if the user requests a u-service in the video or audio format when in a public place, the system then issues an alternative request, which delivers that u-service in the text format instead.

$$getPreAltReq(s, o, download) = getOnAltReq(s, o, download) = \{(PartOfTest(o), download)\} \tag{5.15}$$

for $o \in O_{test}$

This function means that if the user requests a u-test service and if the condition does not hold, the system then issues an alternative request, which in this case means that the system delivers the 'part of test' form.

PreD(s, o, r) = PreAdapted(getPreADAPT(s, o, r)).

This is the adaptation predicate; it takes the form of true or false, which means that if the adaptation process is successful, 'true' will be returned, otherwise 'false' is returned.

The access permission decision is defined as:

$allowed(s, o, r) \Rightarrow preD(s, o, r)$;
This predicate denotes that the subject s is granted the right r to access the object o, if the adaptation process for this access request is successful.

$$ended(s, o, r) \Rightarrow \left(\begin{array}{c} \neg preD(s, o, r) \\ \wedge \\ \bigvee\limits_{(o', r') \in E} allowed(s, o', r') \end{array} \right)$$

This predicate denotes that the right r requested by the subject s to access the object o is ended, if the predicate *preD* is false and an alternative request is issued. where $E = getPreAltReq(s, o, r)$ and the symbol '$\Rightarrow$' denotes the logical implication.

5.4 Formal specification in *CCA*

We now present the formalisation process of the CA-UCON model used in case study above (i.e. in a u-learning system). This formalisation process is completed utilizing the *CCA* notation explained in the previous chapter. We formalise and highlight the important parts of this case study, which are represented by Adaptation and Conditions.

5.4.1 CheckPreC Ambient

This ambient receives the request from *requesting* ambient in order to check the condition of the current request, which in this case is the format of the u-service and the context of the mobile user. Thus, the *checkPreC* ambient senses the context of the user, the memory size of the mobile device and the bandwidth of the network, and then sends the *context*, *fm* and *bandwidth* parameters to the *SubjectCxt*, *MemorySize* and *Bandwidth* ambients, respectively. Then, if the u-lecture is in video or text format and the context of the user is *driving*, and if the memory size of the mobile device is more than 2MB, the *checkPreC* ambient will send the parameter *AltA* to the *requesting* ambient; the latter then sends it to the *preadapting* ambient for adaptation.

However, if the u-lecture is in video or audio format and the context of the user is in a public place, and if the memory size of the mobile device is more than 1MB, the *checkPreC* ambient will send the parameter *AltT* to the *requesting* ambient; the latter then sends it to the *preadapting* ambient for adaptation. If the u-lecture is in video or text format and the context of the user is *driving*, and if the memory size of the mobile device is less than or equal to 2MB, the *checkPreC* ambient will

send the parameter *AltA1* to the *requesting* ambient; the latter then sends it to the *preadapting* ambient for adaptation. If the u-lecture is in video or audio format and the context of the user is in a public place, and if the memory size of the mobile device is less than or equal to 1MB, the *checkPreC* ambient will send the parameter *AltT1* to the *requesting* ambient; the latter then sends it to the *preadapting* ambient for adaptation.

If the u-lecture is in video format and the context of the user is private, and if the memory size of the mobile device is more than 5MB and the bandwidth is low, the *checkPreC* ambient will send the parameter *action1* to the *requesting* ambient; the latter then sends it to the *preadapting* ambient for adaptation. If the u-lecture is in video format and the context of the user is *private*, and if the memory size of the mobile device is less than or equal to 5MB and bandwidth is low, the *checkPreC* ambient will send the parameter *action2* to the *requesting* ambient; the latter then sends it to the *preadapting* ambient for adaptation.

If the u-lecture is in audio format and the context of the user is *private*, and if the memory size of the mobile device is less than or equal to 2MB, the *checkPreC* ambient will send the parameter *action2* to the *requesting* ambient; the latter then sends it to the *preadapting* ambient for adaptation. If the u-lecture is in text format and the context of the user is *private*, and if the memory size of the mobile device is less than or equal to 1MB, the *checkPreC* ambient will send the parameter *action2* to the *requesting* ambient; the latter then sends it to the *preadapting* ambient for adaptation. If the u-lecture is in video format and the context of the user is *private*, and if the memory size of the mobile device is less than or equal to 5MB and bandwidth is low, the *checkPreC* ambient will send the parameter *action3* to the *requesting* ambient; the latter then sends it to the *preadapting* ambient for adaptation. Otherwise, the *checkPreC* ambient will send 1 to the *requesting* ambient; the latter then sends it to the *accessing* ambient. This behaviour is modelled as follows:

$$P_{checkPreC} \;\widehat{=}\; !requesting :: (requesterid, lectFormat, download).subjectCxt \downarrow \langle requesterid \rangle.$$
$$subjectCxt \downarrow (req, context).MemorySize \downarrow \langle requesterid \rangle.MemorySize \downarrow (req, fm).$$
$$Bandwidth \downarrow \langle requesterid \rangle.Bandwidth \downarrow (req, bandwidth).$$

$$\left\{\begin{array}{l}
(lectFormat = video) \wedge (context = driving) \wedge (fm > 2)?requesting :: \langle AltA \rangle.0 \\
|(lectFormat = video) \wedge (context = public) \wedge (fm > 1)?requesting :: \langle AltT \rangle.0 \\
|(lectFormat = video) \wedge (\neg(context = driving)) \vee (\neg(context = piblic)) \wedge (fm > 5) \\
\wedge(bandwidth = high)?requesting :: \langle 1 \rangle.0 \\
|(lectFormat = video) \wedge (context = driving) \wedge (fm <= 2)?requesting :: \langle AltA1 \rangle.0 \\
|(lectFormat = video) \wedge (\neg(context = public) \vee (context = driving)) \wedge (fm <= 5) \\
\wedge(bandwidth = high)?requesting :: \langle Action2 \rangle.0 \\
|(lectFormat = video) \wedge (context = public) \wedge (fm <= 1)?requesting :: \langle AltT1 \rangle.0 \\
|(lectFormat = video) \wedge (\neg(context = public) \vee (context = driving)) \wedge (bandwidth = low) \\
\wedge(fm > 5)?requesting :: \langle Action1 \rangle.0 \\
|(lectFormat = video) \wedge (\neg(context = public)) \vee (context = driving) \wedge (fm <= 5) \wedge \\
(bandwidth = low)?requesting :: \langle Action3 \rangle.0 \\
|(lectFormat = audio) \wedge (context = public) \wedge (fm > 1)?requesting :: \langle AltT \rangle.0 \\
|(lectFormat = audio) \wedge (context = public) \wedge (fm <= 1) >?requesting :: \langle AltT1 \rangle.0 \\
|(lectFormat = audio) \wedge (\neg(context = public)) \wedge (fm > 2)?requesting :: \langle 1 \rangle.0 \\
|(lectFormat = audio) \wedge (\neg(context = public)) \wedge (fm <= 2)?requesting :: \langle Action2 \rangle.0 \\
|(lectFormat = text) \wedge (context = driving) \wedge (fm > 2)?requesting :: \langle AltA \rangle.0 \\
|(lectFormat = text) \wedge (\neg(context = driving)) \wedge (fm > 1)?requesting :: \langle 1 \rangle.0 \\
|(lectFormat = text) \wedge (\neg(context = driving)) \wedge (fm <= 1)?requesting :: \langle Action2 \rangle.0 \\
|(lectFormat = text) \wedge (context = driving) \wedge (fm <= 2)?requesting :: \langle AltA1 \rangle.0
\end{array}\right\} \tag{5.16}$$

5.4.1.1 SubjectCxt Ambient

This ambient receives the parameter *requesterid* from the *checkPreC* ambient; this parameter represents the ID of the subject. The *SubjectCxt* ambient sends the parameters *requesterid* and *context* back to the *checkPreC* ambient, which represent that the ID of the subject and his/her context. We model this ambient as the simulation of a sensor in the real world, which randomly gives the context of subject.

$$P_{subjectCxt} \;\widehat{=}\; !\uparrow(requesterid).\left\{\begin{array}{l}\uparrow\langle requesterid, driving\rangle.0|\\ \uparrow\langle requesterid, private\rangle.0|\\ \uparrow\langle requesterid, Public\rangle.0\end{array}\right\}$$

5.4.1.2 MemorySize Ambient

This ambient receives the parameter *requesterid* from the *checkPreC* ambient; this parameter represents the ID of the mobile user. We use the variable *fm* to denote the free memory of mobile device. The *MemorySize* ambient sends the parameters *requesterid* and *fm* back to the *checkPreC* ambient, which represent the ID of the mobile user and the memory size of his/her device. We model this ambient as the simulation of a sensor in the real world, which gives the memory size of the mobile

device in a random manner.

$$P_{MemorySize} \;\widehat{=}\; !\uparrow(requesterid).\left\{\begin{array}{l}\uparrow\langle requesterid, 4\rangle.0|\\ \uparrow\langle requesterid, 7\rangle.0|\\ \uparrow\langle requesterid, 3\rangle.0|\\ \uparrow\langle requesterid, 8\rangle.0|\\ \uparrow\langle requesterid, 0\rangle.0|\\ \uparrow\langle requesterid, 5\rangle.0\end{array}\right\}$$

5.4.1.3 Bandwidth Ambient

This ambient receives the parameter *requesterid* from the *checkPreC* ambient; this parameter represents the ID of the mobile user. We use the variable *bandwidth* to denote the network bandwidth. The *Bandwidth* ambient sends the parameters *requesterid* and *bandwidth* back to the *checkPreC* ambient, which together represent the ID of the mobile user and the bandwidth of the network being used. We model this ambient as the simulation of a sensor in the real world, which gives the bandwidth of the network being used by the mobile device in a random manner.

$$P_{Bandwidth} \;\widehat{=}\; !\uparrow(requesterid).\left\{\begin{array}{l}\uparrow\langle requesterid, low\rangle.0|\\ \uparrow\langle requesterid, high\rangle.0|\end{array}\right\}$$

5.4.2 GC Ambient

This ambient represents the behaviour of the Garbage Collector software, which is requested from the *preadapting* ambient in order to turn on the *GC* ambient to make more space available in the memory device. The *GC* ambient receives the parameter *Areply* from the *preadapting* ambient, and if *Areply* is *memoryislow*, the *GC* ambient will start working for a duration of *5* seconds. Following this, the *GC* ambient sends the parameter *checkfm* to the *FMem* ambient and receives the parameter *fm* from the *FMem* ambient which is the memory capacity and sends it back to the *preadapting* ambient. This behaviour is modelled as follows:

$$P_{GC} \;\widehat{=}\; \left\{ \begin{array}{l} !preadapting :: (Areply). \\ (Areply = MemoryisLow)\ ?FMem :: \langle checkfm \rangle. \\ FMem :: (fm).perAdapting :: \langle fm \rangle.0 \end{array} \right\} \tag{5.17}$$

5.4.3 preadapting Ambient

This ambient receives the request from the *requesting* ambient, which means that the conditions for this request have not been fulfilled in order to access the service. So, the current request needs to be adapted to the new situation based on the current context of the mobile user. If the reply is *AltA*, the *preadapting* ambient issues an alternative request and sends the u-service in the audio format along with *1* to the *requesting* ambient; this means that the adaptation has been successful. However, if the reply is *AltA1*, the *preadapting* ambient sends the parameter *memoryisLow* to the *GC* ambient in order to increase the memory capacity of the mobile device; it then receives the parameter *fm*, and if *fm* is more than 2MB, the audio format will

be sent along with *1* to the *requesting* ambient, which means that the adaptation has been successful. Otherwise, *0* along with the audio format will be sent to the *requesting* ambient, which means that the adaptation has failed.

However, if the reply is *AltT*, the *preadapting* ambient issues an alternative request and sends the text format along with *1* to the *requesting* ambient, which means that the adaptation has been successful. However, if the reply is *AltT1*, the *preadapting* ambient sends the parameter *memoryisLow* to the *GC* ambient in order to increase the memory capacity of the mobile device; the *preadapting* ambient then receives the parameter *fm*, and if *fm* is more than 1MB, the text format will be sent along with *1* to the *requesting* ambient, which means that the adaptation has been successful. Otherwise, *0* along with the text format will be sent to the *requesting* ambient, which means that the adaptation has failed.

If the reply is *action1*, it means that the bandwidth of the network being used by the mobile device is low; the *preadapting* ambient then sends the parameter *bandwidthislow* to the *HB* ambient in order to switch to a high bandwidth network. If *HBreply* returns with 'high', the video format together with *1* is sent to the *requesting* ambient. Otherwise, *0* is sent to the *requesting* ambient (as a result of adaptation failure). If the reply is *action2*, it means that the memory size of the mobile device is low; the *preadapting* ambient then sends the parameter *memoryislow* to the *GC* ambient in order to increase the capacity of the memory. If *fm* is greater than requested, the requested format together with *1* is sent to the *requesting* ambient. Otherwise, *0* is sent to the *requesting* ambient (as a result of adaptation failure).

If the reply is *action3*, it means that the network bandwidth being used by the mobile device is low and that the memory capacity is low. In this case, the *preadapt-*

ing ambient sends the parameters *bandwidthislow* and *memoryislow* to the *HB* and *GC* ambients, respectively, in order to switch to a high bandwidth network and to increase the memory capacity. If *HBreply* is returned with 'high' and *fm* is greater than requested, the video format together with *1* is sent to the *requesting* ambient. Otherwise, *0* is sent to the *requesting* ambient (as a result of adaptation failure). This behaviour is modelled as follows:

$P_{preadapting} \;\widehat{=}\; !requesting :: (requesterid, lectFormat, download, reply).\{(5.18)|(5.19)\}$

Where:

$$
\left\{
\begin{array}{l}
(reply = AltA)\ ?\ requesting :: \langle 1, audio\rangle.0 \\
|(reply = AltA1)\ ?\ GC :: \langle MemoryisLow\rangle.GC :: (fm). \\
\left\{ \begin{array}{l} (fm > 2)?\ requesting :: \langle 1, audio\rangle.0 \\ |(fm <= 2)?\ requesting :: \langle 0, audio\rangle.0 \end{array} \right\} \\
|(reply = AltT)\ ?\ requesting :: \langle 1, text\rangle.0 \\
|(reply = AltT1)\ ?\ GC :: \langle MemoryisLow\rangle.GC :: (fm). \\
\left\{ \begin{array}{l} (fm <= 1)?\ requesting :: \langle 0, text\rangle.0 \\ |(fm > 1)?\ requesting :: \langle 1, text\rangle.0 \end{array} \right\} \\
|(reply = Action1)?\ HB :: \langle bandwidthislow\rangle.HB :: (HBreply). \\
\left\{ \begin{array}{l} (HBreply = high)?\ requesting :: \langle 1, video\rangle.0 \\ |(HBreply = low)?\ requesting :: \langle 0, video\rangle.0 \end{array} \right\} \\
|(reply = Action2)\ and\ (lectFormat = video)?\ GC :: \langle MemoryisLow\rangle.GC :: (fm). \\
\left\{ \begin{array}{l} (fm > 5)?\ requesting :: \langle 1, video\rangle.0 \\ |(fm <= 5)?\ requesting :: \langle 0, video\rangle.0 \end{array} \right\} \\
|(reply = Action2)\ and\ (lectFormat = audio)?\ GC :: \langle MemoryisLow\rangle.GC :: (fm). \\
\left\{ \begin{array}{l} (fm > 2)?\ requesting :: \langle 1, audio\rangle.0 \\ |(fm <= 2)?\ requesting :: \langle 0, audio\rangle.0 \end{array} \right\}
\end{array}
\right\}
\qquad (5.18)
$$

$$\left\{\begin{array}{l}(reply = Action2)\ and\ (lectFormat = text)?\ GC :: \langle MemoryisLow\rangle.GC :: (fm).\\ \left\{\begin{array}{l}(fm > 1)?\ requesting :: \langle 1, text\rangle.0\\ |(fm <= 1)?\ requesting :: \langle 0, text\rangle.0\end{array}\right\}\\ |(reply = Action3)?\ HB :: \langle bandwidthislow\rangle.HB :: (HBreply).\\ \left\{\begin{array}{l}(not(HBreply = high))?\ requesting :: \langle 0, video\rangle.0\\ |(HBreply = high)?\ GC :: \langle MemoryisLow\rangle.GC :: (fm).\\ \left\{\begin{array}{l}(fm <= 5)?\ requesting :: \langle 0, video\rangle.0\\ |(fm > 5)?\ requesting :: \langle 1, video\rangle.0\end{array}\right\}\end{array}\right\}\end{array}\right\} \tag{5.19}$$

5.4.4 Subject Ambient

This ambient sends the request with the parameters (*requesterid, lectFormat, download*) to the *requesting* ambient in order to access the service. The *requesterid* parameter is used to create a user profile. The *lectFormat* parameter represents the lecture format that the user has requested. The *download* parameter represents the right requested by the user. Thus, the *subject* ambient receives a reply from the *requesting* ambient, which is *permit* if all the requirements have been fulfilled or *deny* if otherwise. The user eventually ends the usage by sending the message END_USAGE to the *accessing* ambient. This behaviour is modelled as follows:

$$\begin{array}{l}P_{subject} \mathrel{\widehat{=}} !requesting :: \langle P1, video, download\rangle.requesting :: (reply, video, download).\{\\ \qquad (reply = PERMIT)?accessing :: (x, o, r).0\\ \quad | \ accessing :: \langle \text{END\_USAGE}, video, download\rangle.0\\ \}\end{array}$$

5.4.5 FMem Ambient

This ambient receives the parameters *reply* from the *GC* ambient to checks the memory capacity of the mobile device. Then, the *FMem* ambient sends the parameter *fm*, which is the memory capacity, back to the *GC* ambient. We model this ambient by using a random function that gives the capacity of the memory in a random manner, as the simulation of a sensor in the real world. This behaviour is modelled as follows:

$$P_{FMem} \;\widehat{=}\; !GC :: (reply).GC :: \langle fm \rangle.0$$

5.4.6 HB Ambient

This ambient represents the behaviour of the network bandwidth. The *HB* ambient receives the parameter *Areply* from the *preadapting* ambient, and if *Areply* is *bandwidthislow*, the *HB* ambient attempts to switch from a low bandwidth to a high bandwidth in the mobile device. We model this ambient using a random function that gives the bandwidth of the network in a random manner, as the simulation of a sensor in the real world. This behaviour is modelled as follows:

$$P_{HB} \;\widehat{=}\; \left\{ \begin{array}{l} !preadapting :: (Areply). \\ (Areply = bandwidthisLow)\ ?preadapting :: \langle HBreply \rangle.0 \end{array} \right\} \quad (5.20)$$

5.5 Validation

In this section, we present the validation of our CA-UCON model vis-a-vis the above case study (u-learning system). We illustrate how CA-UCON model properties can be validated using the execution environment of *CCA* [96]. Firstly, we present the

syntax and the execution environment of ccaPL. We then devise scenarios and execute them in order to validate two main classes of system properties as proposed by [71]:

Safety property: stating that nothing bad will happen.
Liveness property: stating that something good will happen, eventually

For the evaluation of our system we do not use the a metric (benchmark) because the CCA is used to validate the above mentioned properties: Safety property and Liveness property. Therefore, CCA is powerful language which has an execution environment that can easy run the formal specification of mobile and context-aware system to validate its properties. We advise a different scenarios and execute them to validate a various properties of the system.

5.5.1 ccaPL: A Programming Language for CCA

In the previous chapter, we presented the mathematical notation of *CCA*. It is appropriate notation for modelling systems that are mobile and context-aware. Our CA-UCON model is modelled by using the *CCA* notation. In the following subsections the syntax, context expression and execution environment of *ccaPL* are presented.

5.5.1.1 Syntax of ccaPL

Using a normal text editor for reproducing the syntax of *CCA* is difficult because of the special symbols that are used in the syntax of *CCA* (eg.$'\uparrow'$ and $'\downarrow'$). Thus, the syntax of ccaPL is presented in a manner that is readable by the computer. Capabilities, processes and locations are shown in Tables 5.1,5.2, 5.3 below.

Table 5.1: Capabilities of ccaPL

Capabilities M		
CCA	$ccaPL$	Description
`in` n	`in` n	move into ambient n
`out`	`out`	move out
`del` n	`del` n	delete ambient n
$\alpha\ x\langle\tilde{z}\rangle$	$(\alpha\ )x(\tilde{z})$	call to the process abstraction x
$\alpha\ (\tilde{y})$	$(\alpha\ )$`recv`$(\tilde{y})$	receive list of messages $\tilde{y}\ from\ \alpha$
$\alpha\ \langle\tilde{y}\rangle$	$(\alpha\ )$`send`$(\tilde{z})$	send list of messages $\tilde{z}\ to\ \alpha$

Table 5.2: Processes of ccaPL

Processes P		
CCA	$ccaPL$	Description
`0`	`0`	inactivity
$n[P]$	$n[P]$	ambient x
$P\|Q$	$P\|Q$	parallel composition
$!P$	$!(P)$	replication
$(\nu n)\ P$	`new` n (P)	restriction
$\kappa?M.P$	$< k > M.P$	context-guarded capability
$x \triangleright (\tilde{y}).P$	`proc` $x(\tilde{y}).P$	process abstraction x
{P}	{P}	brackets

Table 5.3: Location of ccaPL

Location α		
CCA	$ccaPL$	Description
$\uparrow$	@	any parent
$n\uparrow$	n@	parent n
$\downarrow$	#	any child
$n\downarrow$	n#	child n
::	::	any sibling
n ::	n ::	sibling n
ϵ	ϵ	locally

5.5.1.2 Context Expressions in ccaPL

In Table 5.4 below the context expressions of ccaPL are shown. In ccaPL the non readable symbols which are utilized by CCA are changed to readable one that can be written in normal text editor in order to represent context expression in a usual manner.

Table 5.4: Context Expressions of ccaPL

Context Expressions κ		
CCA	*ccaPL*	Description
T	`true`	true
$n = m$	$n = m$	name match
$\bullet$	`this`	hole
$\neg\kappa$	$not(\kappa)$	negation
$\kappa_1 \vert \kappa_2$	$\kappa_1 \vert \kappa_2$	parallel composition
$\kappa_1 \vee \kappa_2$	κ_1 `or` κ_2	disjunction
$\kappa_1 \wedge \kappa_2$	κ_1 `and` κ_2	conjunction
$\oplus\kappa$	`next` (κ)	spatial next modality
$\diamondsuit\kappa$	`somewhere` (κ)	somewhere modality

5.5.1.3 ccaPL Execution Environment

The architecture of the execution environment of ccaPL is shown in Figure 5.2 below. This architecture consists of the following components: editor, parser, interpreter, manager and console. The editor component represents the development environment of the *CCA* program, which allows the user to write a *CCA* program in a normal text editor. The second component is the parser, which is developed by utilizing JavaCC (Java Compiler Compiler); the responsibility of this component is to verify the syntax of any program that is written in *ccaPL*. The manager component is mange the console and the interpreter components. The next component is the interpreter; the responsibility of this component is the execution of the program in *ccaPL*, based on *CCA* reduction semantics. The last component is the console; this

is considered as part of the execution environment and is currently being created.

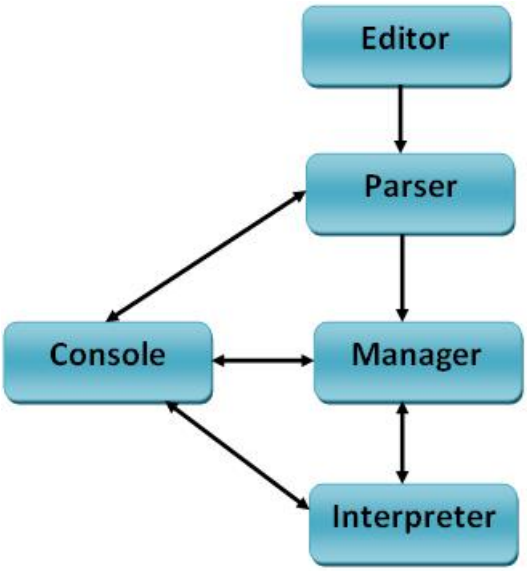


Figure 5.2: The architecture of the execution environment of ccaPL [96]

The comments can be written in ccaPL program using this symbol "//" if it is a single line commentary or between these symbol "/*","*/" if it is multiple lines commentary. So, the JDK (Java Development Kit) has to be installed in computer in order to use the execution environment to execute ccaPL program. The following example below shows how the ccaPL program is written:

```
BEGIN_DECLS
def has(n) = this | n[true] | true
END_DECLS
// Here goes the program
Abdul[::recv(sder,x).<x=hello or x=hi>let y = thanks_+sder in sder::send(y).0]
|
francois[::send(francois, hello).::recv(ack).0]
```

The declaration section of a ccaPL program starts with key word "BEGIN_DECLS" and ending with another key word "END_DECLS", these key words must be writ-

ten in capital letters. This is an optional section and is used to define the context expression.

In order to present how the execution environment of ccaPL works in practice, we will execute the above instance. The Figure 5.3 below shows the result of the execution of the above example.

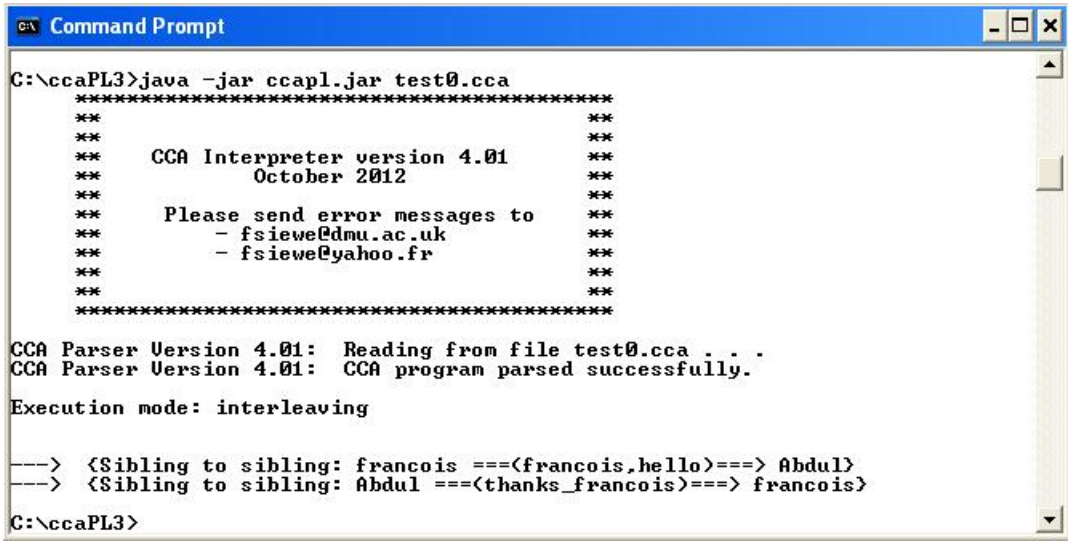


Figure 5.3: Reduction relation of execution environment of ccaPL

Now, the output of an execution will be explained as follows: the meaning of the symbol $'- - - - - >'$ is the reduction relation of CCA as described in [96]. A pair of curly brackets contains the explanation of each transition, which describes the relationship between the ambients, such as "Parent to child", "Child to parent" and "Sibling to sibling". In addition, it specifies the sender and the receiver as well as the massage exchanged between them. For example, the notation C $======(X)====== >$ D represents that an ambient C sent a message 'X' to ambient D.

5.5.2 Executing Scenarios

In following paragraphs, we design a different scenarios and execute them in order to validate properties of CA-UCON model in u-learning system. The two main classes of system properties are safety property and liveness property:

Scenario 1: the property to validate in this scenario is : *" if the pre-authorization or the pre-obligation requirement is not met before the access, then the access request will be denied by the system"*. Suppose a learner requests to download a u-lecture in the video format and the context of learner is in *private* place and the memory capacity of her/his mobile device is more than 5MB. However, the pre-obligation requirements of this access request are not met. In this case, the system has to deny the access request.

```
**            October 2012              **
**                                      **
**     Please send error messages to    **
**         - fsiewe@dmu.ac.uk           **
**         - fsiewe@yahoo.fr            **
**                                      **
**                                      **
******************************************

CCA Parser Version 4.01:  Reading from file ca_ucon6.1.cca . . .
CCA Parser Version 4.01:  CCA program parsed successfully.
Execution mode: interleaving

1- --->  {local call to the abstraction "mem" in the ambient "root"}
2- --->  {Sibling to sibling: subject ===(P1,video,download)===> requesting}
3- --->  {Sibling to sibling: requesting ===(P1,video,download)===> checkpreA}
4- --->  {Sibling to sibling: checkpreA ===(1)===> requesting}
5- --->  {Sibling to sibling: requesting ===(P1,video,download)===> checkpreB}
6- --->  {Sibling to sibling: checkpreB ===(P1)===> UP}
7- --->  {Sibling to sibling: UP ===(0)===> checkpreB}
8- --->  {Sibling to sibling: checkpreB ===(0)===> requesting}
9- --->  {Sibling to sibling: requesting ===(denyB,video,download)===> subject}
```

Figure 5.4: Execution of Scenario 1

Figure 5.4 illustrates the execution of scenario 1. The subject sends the access request containing the subject ID, the u-lecture in the video format and the downloading right to the *requesting* ambient (line 2). The *requesting* ambient receives the access request and checks the *pre-obligation* pertaining to this request (line 3). So, the *pre-obligation* of this request is not met, and finally the system denied the access request (lines 4-5). So, it can be seen from this scenario the safety property is validated because the system deny the request if the pre-obligation is not met.

Scenario 2: The property to validate in this scenario is : *" if the pre-authorization, pre-obligation and pre-condition requirements are met at the time of access request and on-authorization, on-obligation and on-condition requirements continuously hold during the access, the access will end successfully"*. Suppose a learner requests to download a u-lecture in the video format and the context of learner is in *private* place and the memory capacity of her/his mobile device is more than 5MB. The

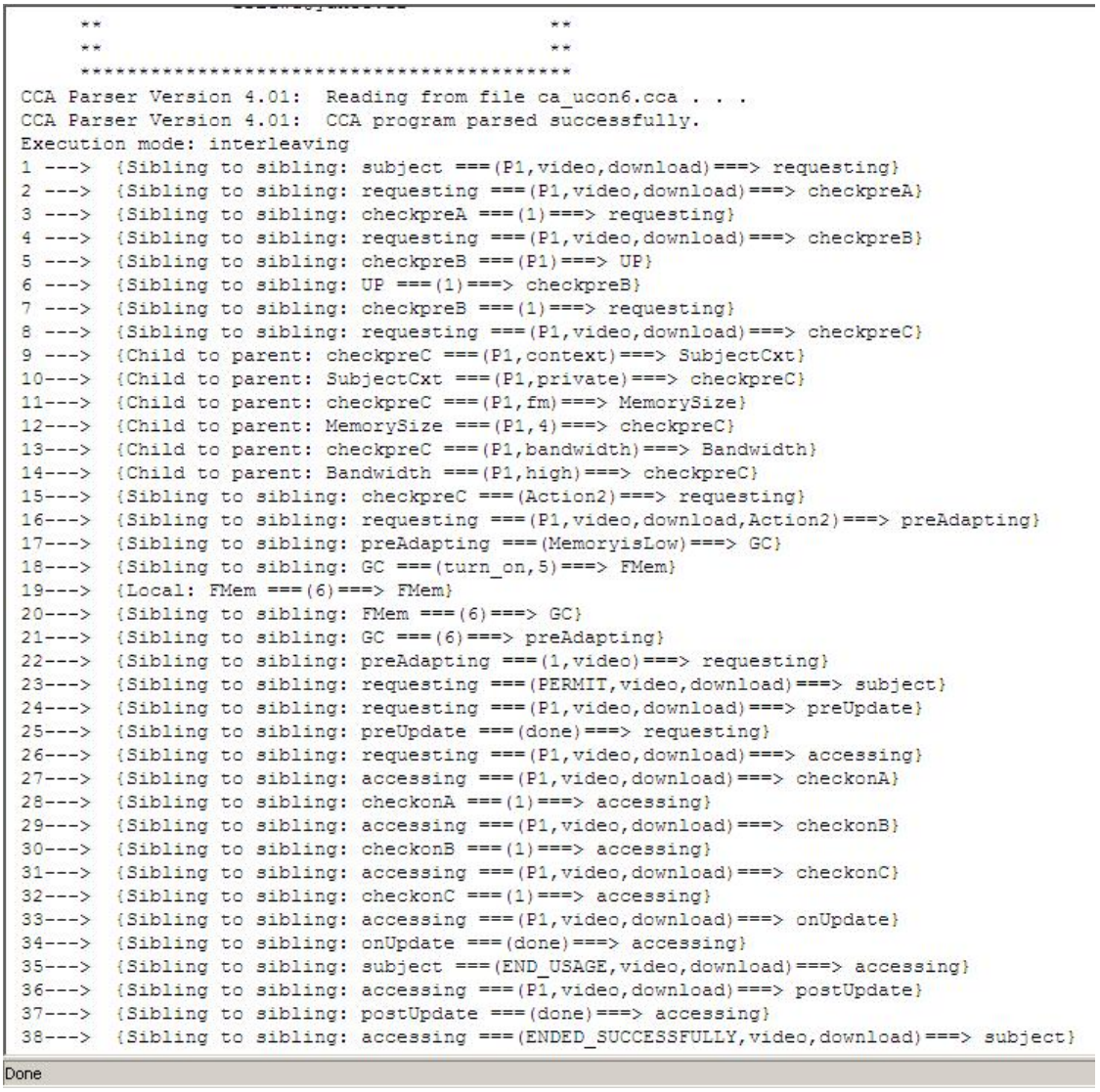
service will be delivered to the learner by the system.

```
      **                                    **
      **                                    **
      ****************************************
CCA Parser Version 4.01:  Reading from file ca_ucon6.cca . . .
CCA Parser Version 4.01:  CCA program parsed successfully.
Execution mode: interleaving
1 --->  {Sibling to sibling: subject ===(P1,video,download)===> requesting}
2 --->  {Sibling to sibling: requesting ===(P1,video,download)===> checkpreA}
3 --->  {Sibling to sibling: checkpreA ===(1)===> requesting}
4 --->  {Sibling to sibling: requesting ===(P1,video,download)===> checkpreB}
5 --->  {Sibling to sibling: checkpreB ===(P1)===> UP}
6 --->  {Sibling to sibling: UP ===(1)===> checkpreB}
7 --->  {Sibling to sibling: checkpreB ===(1)===> requesting}
8 --->  {Sibling to sibling: requesting ===(P1,video,download)===> checkpreC}
9 --->  {Child to parent: checkpreC ===(P1,context)===> SubjectCxt}
10--->  {Child to parent: SubjectCxt ===(P1,private)===> checkpreC}
11--->  {Child to parent: checkpreC ===(P1,fm)===> MemorySize}
12--->  {Child to parent: MemorySize ===(P1,6)===> checkpreC}
13--->  {Child to parent: checkpreC ===(P1,bandwidth)===> Bandwidth}
14--->  {Child to parent: Bandwidth ===(P1,high)===> checkpreC}
15--->  {Sibling to sibling: checkpreC ===(1)===> requesting}
16--->  {Sibling to sibling: requesting ===(PERMIT,video,download)===> subject}
17--->  {Sibling to sibling: requesting ===(P1,video,download)===> preUpdate}
18--->  {Sibling to sibling: preUpdate ===(5)===> francois_credit}
19--->  {Sibling to sibling: francois_credit ===()===> preUpdate}
20--->  {Sibling to sibling: preUpdate ===(done)===> requesting}
21--->  {Sibling to sibling: requesting ===(P1,video,download)===> accessing}
22--->  {Sibling to sibling: accessing ===(P1,video,download)===> checkonA}
23--->  {Sibling to sibling: checkonA ===(1)===> accessing}
24--->  {Sibling to sibling: accessing ===(P1,video,download)===> checkonB}
25--->  {Sibling to sibling: checkonB ===(1)===> accessing}
26--->  {Sibling to sibling: accessing ===(P1,video,download)===> checkonC}
27--->  {Sibling to sibling: checkonC ===(1)===> accessing}
28--->  {Sibling to sibling: accessing ===(P1,video,download)===> onUpdate}
29--->  {Sibling to sibling: onUpdate ===(1)===> francois_credit}
30--->  {Local: francois_credit ===(5)===> francois_credit}
31--->  {Sibling to sibling: francois_credit ===()===> onUpdate}
32--->  {Sibling to sibling: onUpdate ===(done)===> accessing}
33--->  {Sibling to sibling: subject ===(END_USAGE,video,download)===> accessing}
34--->  {Sibling to sibling: accessing ===(P1,video,download)===> postUpdate}
35--->  {Sibling to sibling: postUpdate ===(0)===> francois_credit}
36--->  {Sibling to sibling: francois_credit ===()===> postUpdate}
37--->  {Sibling to sibling: postUpdate ===(done)===> accessing}
38--->  {Sibling to sibling: accessing ===(ENDED_SUCCESSFULLY,video,download)===> subject}
Done
```

Figure 5.5: Execution of Scenario2

Figure 5.5 presents a screen shot of the execution of Scenario 2, from which it can be seen that the subject sends the access request to the *requesting* ambient (line 2). The *requesting* ambient receives the access request and checks the *pre-authorisation, pre-obligation* and *pre-conditions* pertaining to this request (lines 3-8). The ambient *checkPreC* checks the context of the subject, the memory capacity of the mobile device and the bandwidth of the network before access. In this case, all the requirements before the access are met and the access is *permit* to the service(lines 16). Moreover, the *requesting* ambient sends the access request to *accessing* ambient to check *on-authorisation, on-obligation* and *on-conditions* pertaining to this request, and all requirements are hold during the access (lines 22-27). In line(33) The subject ended the access successfully by sending this message END_USAGE to the *accessing* ambient.

Scenario 3: The property we want to validate in this scenario is : *" if the pre-authorization and pre-obligation requirements are met, but the pre-condition requirements are not met and the pre-adaptation is successful, eventually the access request will be permit"*. Suppose a learner sends a request to download a u-lecture in video format and the context of learner is in *private* place and the memory capacity of mobile device is less than 5MB. One condition for this service is that the available memory in the mobile device must be more than 5MB. This condition is not met by the current context, so the system will turn on a garbage collector (GC) in an attempt to generate sufficient space to allow the downloading of the service to take place.



```
     **                                    **
     **                                    **
     ****************************************
CCA Parser Version 4.01:  Reading from file ca_ucon6.cca . . .
CCA Parser Version 4.01:  CCA program parsed successfully.
Execution mode: interleaving
1 --->  {Sibling to sibling: subject ===(P1,video,download)===> requesting}
2 --->  {Sibling to sibling: requesting ===(P1,video,download)===> checkpreA}
3 --->  {Sibling to sibling: checkpreA ===(1)===> requesting}
4 --->  {Sibling to sibling: requesting ===(P1,video,download)===> checkpreB}
5 --->  {Sibling to sibling: checkpreB ===(P1)===> UP}
6 --->  {Sibling to sibling: UP ===(1)===> checkpreB}
7 --->  {Sibling to sibling: checkpreB ===(1)===> requesting}
8 --->  {Sibling to sibling: requesting ===(P1,video,download)===> checkpreC}
9 --->  {Child to parent: checkpreC ===(P1,context)===> SubjectCxt}
10--->  {Child to parent: SubjectCxt ===(P1,private)===> checkpreC}
11--->  {Child to parent: checkpreC ===(P1,fm)===> MemorySize}
12--->  {Child to parent: MemorySize ===(P1,4)===> checkpreC}
13--->  {Child to parent: checkpreC ===(P1,bandwidth)===> Bandwidth}
14--->  {Child to parent: Bandwidth ===(P1,high)===> checkpreC}
15--->  {Sibling to sibling: checkpreC ===(Action2)===> requesting}
16--->  {Sibling to sibling: requesting ===(P1,video,download,Action2)===> preAdapting}
17--->  {Sibling to sibling: preAdapting ===(MemoryisLow)===> GC}
18--->  {Sibling to sibling: GC ===(turn_on,5)===> FMem}
19--->  {Local: FMem ===(6)===> FMem}
20--->  {Sibling to sibling: FMem ===(6)===> GC}
21--->  {Sibling to sibling: GC ===(6)===> preAdapting}
22--->  {Sibling to sibling: preAdapting ===(1,video)===> requesting}
23--->  {Sibling to sibling: requesting ===(PERMIT,video,download)===> subject}
24--->  {Sibling to sibling: requesting ===(P1,video,download)===> preUpdate}
25--->  {Sibling to sibling: preUpdate ===(done)===> requesting}
26--->  {Sibling to sibling: requesting ===(P1,video,download)===> accessing}
27--->  {Sibling to sibling: accessing ===(P1,video,download)===> checkonA}
28--->  {Sibling to sibling: checkonA ===(1)===> accessing}
29--->  {Sibling to sibling: accessing ===(P1,video,download)===> checkonB}
30--->  {Sibling to sibling: checkonB ===(1)===> accessing}
31--->  {Sibling to sibling: accessing ===(P1,video,download)===> checkonC}
32--->  {Sibling to sibling: checkonC ===(1)===> accessing}
33--->  {Sibling to sibling: accessing ===(P1,video,download)===> onUpdate}
34--->  {Sibling to sibling: onUpdate ===(done)===> accessing}
35--->  {Sibling to sibling: subject ===(END_USAGE,video,download)===> accessing}
36--->  {Sibling to sibling: accessing ===(P1,video,download)===> postUpdate}
37--->  {Sibling to sibling: postUpdate ===(done)===> accessing}
38--->  {Sibling to sibling: accessing ===(ENDED_SUCCESSFULLY,video,download)===> subject}
Done
```

Figure 5.6: Execution of Scenario 3

Figure 5.6 illustrates that the execution of Scenario 3. The subject sends the access request containing the subject ID, the u-lecture in the video format and the downloading right to the *requesting* ambient (line 1), whereupon the *requesting* ambient receives the access request and then checks the *pre-authorisation, pre-obligation* and *pre-conditions* pertaining to this request (lines 3-8). The ambient *checkPreC* checks the context of the subject, the memory capacity of the mobile device and the bandwidth of network (lines 9-14). In this case, the available memory on the mobile device is less than 5MB (line 12), which means that the system has to adapt to a

new situation. A garbage collector is activated to free more memory space (lines 15-20). The *pre-adaptation* is successful and the available memory space is increased to 6MB as shown in (line 21) and the video format is then delivered (line 24). So, it can be seen from this scenario the liveness property is validated because the system adapts to new situation in order to continue the access for the user.

Scenario 4: The property to validate in this scenario is : *" if the pre-authorization and the pre-obligation requirement is met, but the pre-condition requirement is not met and the pre-adaptation fails, eventually the access request will be denied"*. Suppose a learner requests a u-tutorial in the video format in a private place and that the memory capacity of the mobile device is more than 5MB. However, she/he is using a low bandwidth network in attempting to download the video, which means that the condition to download this service is not met. Therefore, the system adapts to this new situation by switching the mobile device to a high bandwidth network in order to download this service.

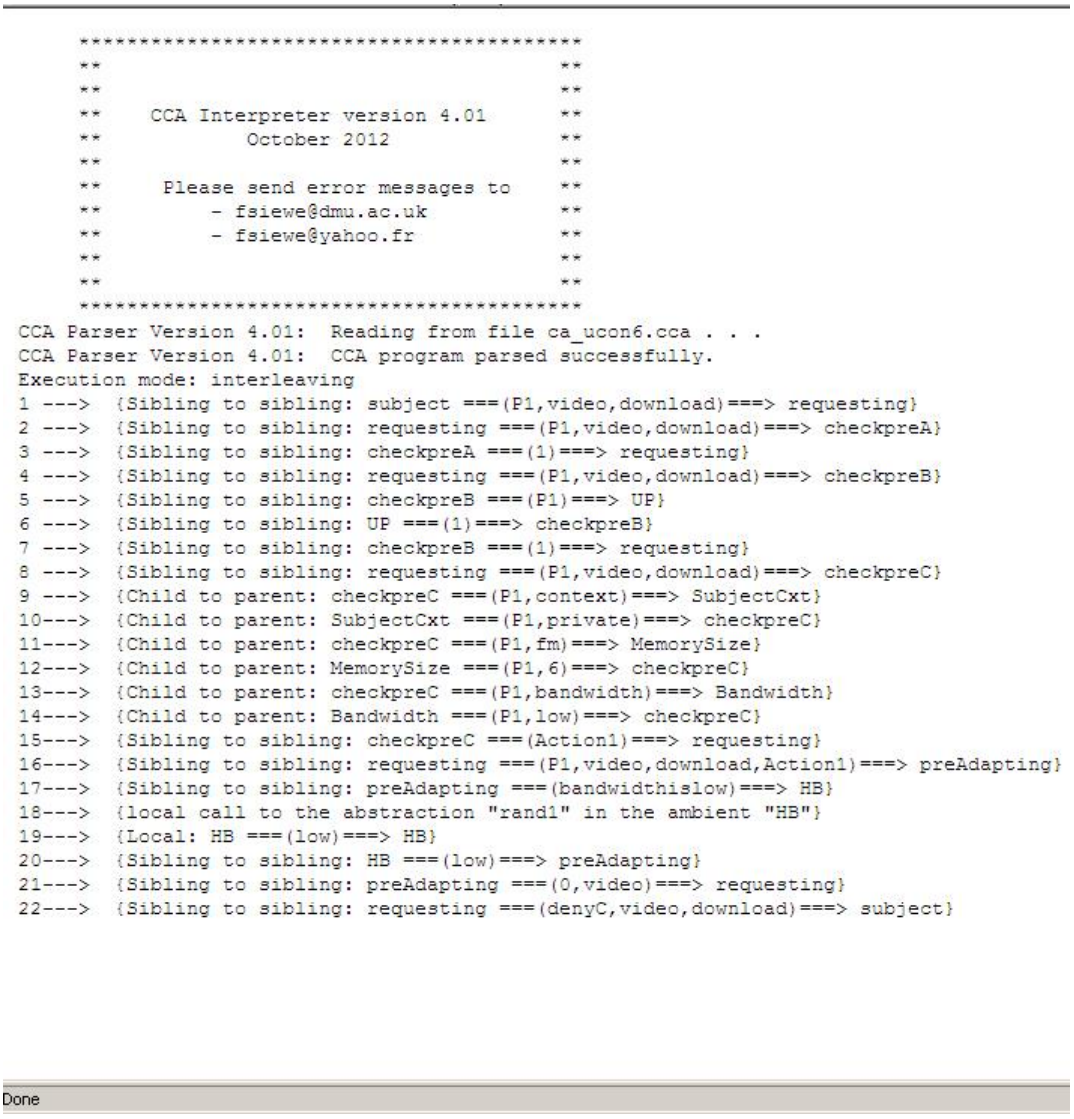


Figure 5.7: Execution of Scenario 4

The execution of Scenario 4 is illustrated in Figure 5.7 the subject sends the access request containing the subject ID, the u-lecture in the video format and the downloading right to the *requesting* ambient (line 1). The *requesting* ambient receives the access request and then checks the *pre-authorization, pre-obligation* and *pre-conditions* pertaining to this request (lines 3-8). The ambient *checkPreC* checks the context of the subject, the memory capacity of the mobile device and the bandwidth of the network (lines 11-14). In this scenario, the bandwidth of the network is low (line 14), so the system has adapt to this new situation by sending

the parameter *Action1* with the request to the *pre-adapting* ambient (line 16) in order to switch to a high bandwidth. Then, the *pre-adapting* ambient receives the current bandwidth of the network from the *HB* ambient is low bandwidth which the adaptation is fail(line 20). Finally, the system is deny the service(line 22).

Scenario 5: In this scenario the property will be validated is : *" if the pre-authorization, pre-obligation and pre-condition requirements are met, but the on-authorization or on-obligation requirements fails, the access will be revoked by the system".* Suppose a learner sends a request to download a u-tutorial in the text format to her/his mobile device when s/he in a public place and that the memory capacity of the device is more than 1MB. However, the on-authorization requirements are not hold, the system will revoke the access.

Figure 5.8: Execution of Scenario 5

Figure 5.8 illustrates the execution of Scenario 5: the subject sends the access request containing the subject ID, the u-tutorial in the text format and the downloading right to the *requesting* ambient (line 1). The *requesting* ambient receives the access request and then checks the *pre-authorisation, pre-obligation* and *pre-conditions* pertaining to this request (lines 2-8). The ambient *checkPreC* checks the context of the subject, the memory capacity of the mobile device and the bandwidth of the network (lines 9-15). So, in this case all requirements before the access are met and the access request is *permit* (line 16). However, the system checks *on-*

authorization during the access and it is not hold (lines 24-25). Finally, the system revokes the access (line 26).

Scenario 6: the property to validate in this scenario is : *" if the on-condition requirements do not hold during the access and the on-adaptation is successful, the access will continue".* Suppose a learner requests a u-tutorial in the video format in a private place and that the memory capacity of the mobile device is more than 5MB and using a high bandwidth at the time of request which means that all *pre-condition* requirements are met by current context. When the system gives the permission to the learner in order to download the video format, she/he starts using a low bandwidth network in attempting to download the video, which means that the on-condition requirements to download this service are not hold based on u-learning system policies. Therefore, the system adapts to this new situation by switching the mobile device to a high bandwidth network in order to download this service.

Figure 5.9 illustrates the execution of Scenario 6: the subject sends the access request containing the subject ID, the u-tutorial in the video format and the downloading right to the *requesting* ambient (line 1). The *requesting* ambient receives the access request and then checks the *pre-authorisation, pre-obligation* and *pre-conditions* requirements pertaining to this request (lines 2-15). Here, the *pre-authorisation, pre-obligation* and *pre-conditions* requirements are met and system gives the permission to the learner to download the service (line 16). The *accessing* ambient receives the access request and then checks the *on-authorisation, on-obligation* and *on-condition* requirements pertaining to this request during the access line(19-30). As the bandwidth is low during the access(line 30), and it is not allowed to access the video format in this context, the system has to adapt to a new situation; it does so by sending the parameter *Action1* with an access request to the *onadapting* ambient (lines 31-35) in order to switch to a high bandwidth.

```
CCA Parser Version 4.01:  Reading from file ca_ucon6.1.cca . . .
CCA Parser Version 4.01:  CCA program parsed successfully.
Execution mode: interleaving
1 --->  {Sibling to sibling: subject ===(P1,video,download)===> requesting}
2 --->  {Sibling to sibling: requesting ===(P1,video,download)===> checkpreA}
3 --->  {Sibling to sibling: checkpreA ===(1)===> requesting}
4 --->  {Sibling to sibling: requesting ===(P1,video,download)===> checkpreB}
5 --->  {Sibling to sibling: checkpreB ===(P1)===> UP}
6 --->  {Sibling to sibling: UP ===(1)===> checkpreB}
7 --->  {Sibling to sibling: checkpreB ===(1)===> requesting}
8 --->  {Sibling to sibling: requesting ===(P1,video,download)===> checkpreC}
9 --->  {Child to parent: checkpreC ===(P1,context)===> SubjectCxt}
10--->  {Child to parent: SubjectCxt ===(P1,private)===> checkpreC}
11--->  {Child to parent: checkpreC ===(P1,fm)===> MemorySize}
12--->  {Child to parent: MemorySize ===(P1,6)===> checkpreC}
13--->  {Child to parent: checkpreC ===(P1,bandwidth)===> Bandwidth}
14--->  {Child to parent: Bandwidth ===(P1,high)===> checkpreC}
15--->  {Sibling to sibling: checkpreC ===(1)===> requesting}
16--->  {Sibling to sibling: requesting ===(PERMIT,video,download)===> subject}
17--->  {Sibling to sibling: requesting ===(P1,video,download)===> preUpdate}
18--->  {Sibling to sibling: preUpdate ===(done)===> requesting}
19--->  {Sibling to sibling: requesting ===(P1,video,download)===> accessing}
20--->  {Sibling to sibling: accessing ===(P1,video,download)===> checkonA}
21--->  {Sibling to sibling: checkonA ===(1)===> accessing}
22--->  {Sibling to sibling: accessing ===(P1,video,download)===> checkonB}
23--->  {Sibling to sibling: checkonB ===(1)===> accessing}
24--->  {Sibling to sibling: accessing ===(P1,video,download)===> checkonC}
25--->  {Child to parent: checkonC ===(P1,context)===> SubjectCxt}
26--->  {Child to parent: SubjectCxt ===(P1,private)===> checkonC}
27--->  {Child to parent: checkonC ===(P1,fm)===> MemorySize}
28--->  {Child to parent: MemorySize ===(P1,6)===> checkonC}
29--->  {Child to parent: checkonC ===(P1,bandwidth)===> Bandwidth}
30--->  {Child to parent: Bandwidth ===(P1,low)===> checkonC}
31--->  {Sibling to sibling: checkonC ===(Action1)===> accessing}
32--->  {Sibling to sibling: accessing ===(P1,video,download,Action1)===> onAdapting}
33--->  {Sibling to sibling: onAdapting ===(bandwidthislow)===> HB}
34--->  {Sibling to sibling: HB ===(high)===> onAdapting}
35--->  {Sibling to sibling: onAdapting ===(1,video)===> accessing}
36--->  {Sibling to sibling: accessing ===(P1,video,download)===> onUpdate}
37--->  {Sibling to sibling: onUpdate ===(done)===> accessing}
38--->  {Sibling to sibling: accessing ===(Continue,video,download)===> subject}
39--->  {Sibling to sibling: subject ===(END_USAGE,video,download)===> accessing}
40--->  {Sibling to sibling: accessing ===(P1,video,download)===> postUpdate}
42--->  {Sibling to sibling: postUpdate ===(done)===> accessing}
43--->  {Sibling to sibling: accessing ===(ENDED SUCCESSFULLY,video,download)===> subject}
Done
```

Figure 5.9: Execution of Scenario 6

Then, the *onadapting* ambient receives the current bandwidth of the network from the *HB* ambient, which becomes a high one (line 34). This means the on-adaptation is successful and the access is continue (line 35).

Scenario 7: In this scenario the property to validate is *" if the on-condition requirements do not hold during the access and the on-adaptation fails, the access will be revoked".* Suppose a learner requests a u-lecture in the video format in a private place and that the memory capacity of the mobile device is more than 5MB and using a high bandwidth at the time of request which means that all *pre-condition* requirements are met by current context. When the system gives the permission to the learner in order to download the video format, she/he starts using a low bandwidth network in attempting to download the video, which means that the *on-condition* requirements to download this service are not hold based on u-learning system policies. Therefore, the system adapts to this new situation by switching the mobile device to a high bandwidth network in order to download this service.

Figure 5.10 illustrates the execution of Scenario 7: the subject sends the access request containing the subject ID, the u-lecture in the video format and the downloading right to the *requesting* ambient (line 1). The *requesting* ambient receives the access request and then checks the *pre-authorisation, pre-obligation* and *pre-conditions* requirements pertaining to this request (lines 2-15). Here, the *pre-authorisation, pre-obligation* and *pre-conditions* requirements are met and system gives the permission to the learner to download the service (line 16). The *accessing* ambient receives the access request and then checks the *on-authorisation, on-obligation* and *on-condition* requirements pertaining to this request during the access line(19-24). As the bandwidth is low during the access(line 30), and it is not allowed to access the video format in this context, the system has to adapt to a new situation; it does so by sending the parameter *Action1* with an access request

```
      **         - fsiewe@dmu.ac.uk           **
      **         - fsiewe@yahoo.fr            **
      **                                      **
      **                                      **
      ******************************************
CCA Parser Version 4.01:  Reading from file ca_ucon6.1.cca . . .
CCA Parser Version 4.01:  CCA program parsed successfully.
Execution mode: interleaving
1 --->  {Sibling to sibling: subject ===(P1,video,download)===> requesting}
2 --->  {Sibling to sibling: requesting ===(P1,video,download)===> checkpreA}
3 --->  {Sibling to sibling: checkpreA ===(1)===> requesting}
4 --->  {Sibling to sibling: requesting ===(P1,video,download)===> checkpreB}
5 --->  {Sibling to sibling: checkpreB ===(P1)===> UP}
6 --->  {Sibling to sibling: UP ===(1)===> checkpreB}
7 --->  {Sibling to sibling: checkpreB ===(1)===> requesting}
8 --->  {Sibling to sibling: requesting ===(P1,video,download)===> checkpreC}
9 --->  {Child to parent: checkpreC ===(P1,context)===> SubjectCxt}
10--->  {Child to parent: SubjectCxt ===(P1,private)===> checkpreC}
11--->  {Child to parent: checkpreC ===(P1,fm)===> MemorySize}
12--->  {Child to parent: MemorySize ===(P1,6)===> checkpreC}
13--->  {Child to parent: checkpreC ===(P1,bandwidth)===> Bandwidth}
14--->  {Child to parent: Bandwidth ===(P1,high)===> checkpreC}
15--->  {Sibling to sibling: checkpreC ===(1)===> requesting}
16--->  {Sibling to sibling: requesting ===(PERMIT,video,download)===> subject}
17--->  {Sibling to sibling: requesting ===(P1,video,download)===> preUpdate}
18--->  {Sibling to sibling: preUpdate ===(done)===> requesting}
19--->  {Sibling to sibling: requesting ===(P1,video,download)===> accessing}
20--->  {Sibling to sibling: accessing ===(P1,video,download)===> checkonA}
21--->  {Sibling to sibling: checkonA ===(1)===> accessing}
22--->  {Sibling to sibling: accessing ===(P1,video,download)===> checkonB}
23--->  {Sibling to sibling: checkonB ===(1)===> accessing}
24--->  {Sibling to sibling: accessing ===(P1,video,download)===> checkonC}
25--->  {Child to parent: checkonC ===(P1,context)===> SubjectCxt}
26--->  {Child to parent: SubjectCxt ===(P1,private)===> checkonC}
27--->  {Child to parent: checkonC ===(P1,fm)===> MemorySize}
28--->  {Child to parent: MemorySize ===(P1,6)===> checkonC}
29--->  {Child to parent: checkonC ===(P1,bandwidth)===> Bandwidth}
30--->  {Child to parent: Bandwidth ===(P1,low)===> checkonC}
31--->  {Sibling to sibling: checkonC ===(Action1)===> accessing}
32--->  {Sibling to sibling: accessing ===(P1,video,download,Action1)===> onAdapting}
33--->  {Sibling to sibling: onAdapting ===(bandwidthislow)===> HB}
34--->  {Sibling to sibling: HB ===(low)===> onAdapting}
35--->  {Sibling to sibling: onAdapting ===(0,video)===> accessing}
36--->  {Sibling to sibling: accessing ===(revokeC,video,download)===> subject}
Done
```

Figure 5.10: Execution of Scenario 7

to the *onadapting* ambient (lines 31-34) in order to switch to a high bandwidth. Then, the *onadapting* ambient receives the current bandwidth of the network from the *HB* ambient, which still low (line 34). This means the on-adaptation is fail and the access is revoked (line 35-36).

5.6 Summary

In this chapter, we have demonstrated the evaluation of CA-UCON model via real world case study of ubiquitous learning system (u-learning system). We illustrate how CA-UCON model properties can be validated using the execution environment of *CCA*. We proposed some scenarios and run them in order to validate two main property: safety and liveness properties.

In the following chapter 6, the architecture of CA-UCON reference monitor is presented and the different types of enforcement architectures for CA-UCON model are demonstrated.

Chapter 6

Enforcement of CA-UCON model

Objectives:

- Present Architecture of CA-UCON Reference Monitor.
- Present Enforcement Architectures of CA-UCON Model.
- Show different examples for enforcement architectures

6.1 Introduction

In this chapter, we demonstrate how the CA-UCON model will be enforced in a system. Firstly, we propose the architecture of the CA-UCON reference monitor (CA-UCON RM) and explain its main components. Secondly, we analyse the enforcement architectures for the CA-UCON model by discussing the three approaches, namely: Centralised Enforcement Architecture, Distributed Enforcement Architecture and Hybrid Enforcement Architecture. Finally, we show the difference between these architectures in terms of enforcement of the CA-UCON RM, using examples.

6.2 Architecture of CA-UCON Reference Monitor

There have been some works conducted in term of usage control enforcement. In spite of the short period of time from the time when the notion of usage control was introduced, there are a lot of substantial efforts in order to create a appropriate enforcement mechanisms. For instance, [87] proposed a new approach known as a Trusted Reference Monitor (TRM in short) which is placed in client side in order to enforce policies. So, the protocol that is used between TRM and application, as well as between different TRMs is relied on challenge-response. Any access request from receiver is followed by a challenge. The requester consequently confirms the application, platform or environment throughout the means of a digital signature. Another approach was proposed by [1] which is similar strategy to the above approach, it is based on a hardware-based Trusted Platform Module (TPM in short) in order to enforce usage policies on digital objects. It is known as self-enforcing objects (SEOs in short) which is used as secure container for transferring objects and poli-

cies and also has the ability to enforce the attached policies on any trusted platform autonomously. Moreover, a similar enforcement approach was proposed by [66] to explain the notion of platform attestation. The system ensures via a WS-Attestation procedure that receiving platform must act correctly before the information is released. Hardware UCON Engine (HUE in short) is another enforcement approach proposed by [61] which is a different from the above mentioned approaches. HUE is considered as secure co-processor with a designated software stack in order to offer integration with the operating system. Next, the RightEnforcer was proposed by [3] which known as a product that is used to enforce simple usage control restrictions i.e. limiting the capability to view, copy, print and store. This approach is incorporated in an e-mail user, so whenever a usage controlled object is delivered to a receiver, the RightEnforcer encrypts the content and then sends the term of use to a centralised RightServer. Then the receiver is forced to use the RightEnforcer in order to be able to decrypt the content and so the policy is always enforced. However, Majority of these enforcement approaches have been done in usage control model (UCON).

Unlike the above UCON enforcement approaches, we propose an architecture of the CA-UCON reference monitor (CA-UCON RM, in short) as depicted in Fig. 6.1, where computing components are represented in a rectangular shape, data storage in a cylindrical shape, and arrows indicate interactions between components. In the following subsections, we explain in detail each component along with its role in this architecture.

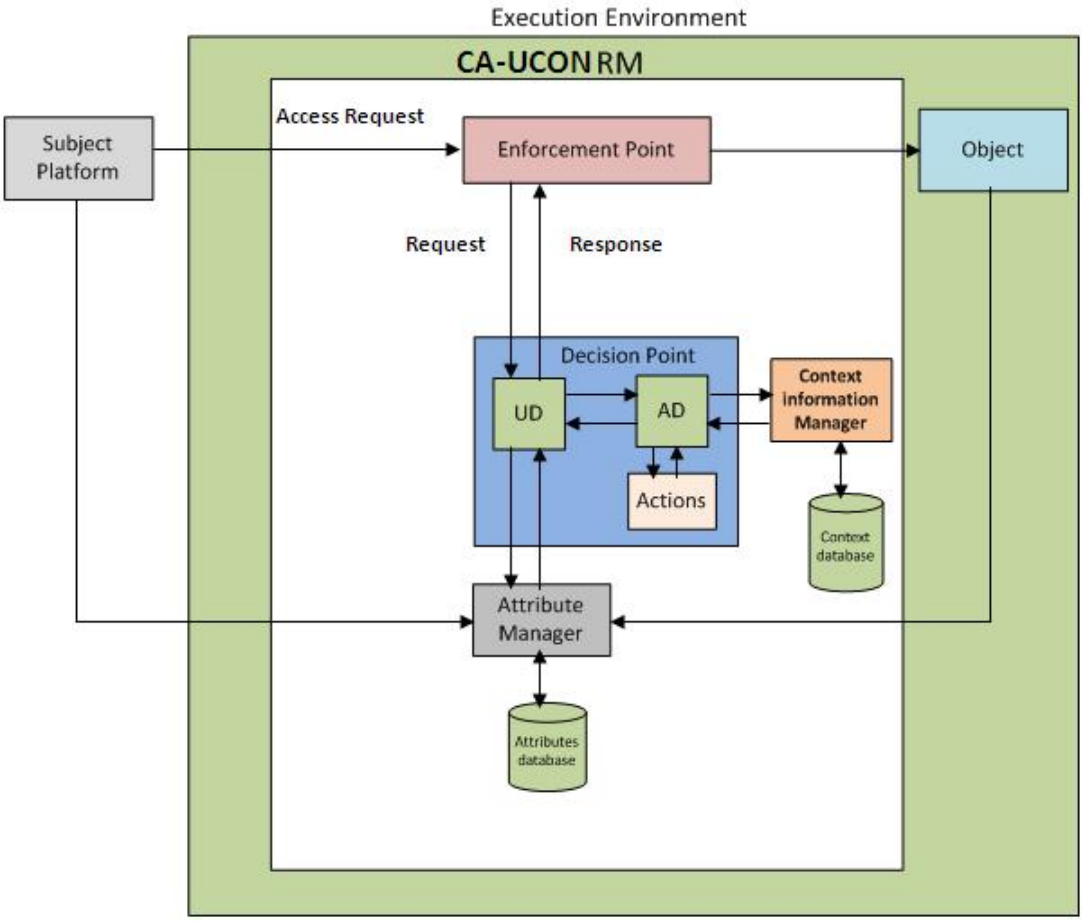


Figure 6.1: Architecture of the CA-UCON Reference Monitor

6.2.1 Enforcement Point

The role of this component is to receive requests from the subject that wishes to access an object and to issue decision requests to the *decision point* component in particular to the usage decision (UD) component and to wait for the response from it, whether PERMIT or DENY. It then lets the subject access the service if the reply from decision point component is PERMIT; otherwise, it denies access to the subject. This behaviour is typical to many access control models [84, 90].

6.2.2 decision point

Unlike in other access control models [84, 90], this component consists of two parts, usage decision (UD) component and adaptation decision (AD) component. The responsibility of the UD component is the evaluation of each access request that it receives from enforcement point component. UD checks the authorization, obligation and condition requirements of each request in order to permit or deny the access request. To do so, UD communicates with the *attribute manager* component in order to acquire the values of the attributes of the subject, object and environment. If all the requirements are met, UD responds by PERMIT to the enforcement point component to allow the access to take place.

However, if the authorization and obligation requirements are met, but the condition requirement is not fulfilled by the environment, UD communicates with the AD component in order to adapt to the new situation. AD interacts with the *context information manager* component to identify the current context of the subject, object and environment. It then performs appropriate actions in an attempt to make the environment meet the condition requirement. This attempt will last for a specified period of time after which the adaptation will be deemed successful or unsuccessful. If successful, the access request will be granted, otherwise it will be denied.

6.2.3 Attribute Manager

The responsibility to this component is to provide access to the database of subject and object attributes in order to use them in access decision. All subject and object attributes will be stored in a database and updated as they change.

6.2.4 Context Information Manager

This component is responsible for monitoring and evaluating the context information such as subject context, object context and environment context. It uses a variety of sensors to sense different types of contexts. It communicates with AD component in order to provide the current context of the subject, object and environment.

6.3 Enforcement Architectures of CA-UCON Model

In this section, we present three different enforcement approaches in order to demonstrate how the CA-UCON model can be implemented in real world situations. The CA-UCON model can be enforced in a number of ways depending on the application. These enforcement approaches are known as the centralized approach, the distributed approach and the hybrid approach. The following subsection will explain each approach in detail to show the difference between these approaches, their usages, and possible benefits/drawbacks.

6.3.1 Centralized Enforcement Architecture

A centralized architecture is dependent upon one node being designated as the computer node or server. This node executes and runs the complete application locally and all users share this central system. Hence control and failure is concentrated in a single place, the server. The CA-UCON RM will be implemented in the server side only. This entails control of the services being the responsibility of the server alone as shown in Figure 6.2.

How does this model support adaptation? The server is installed with a variety of sensors which are able to record the context and behaviour of the subject, object and environment. The adaptation to the new situation and sending of the appropriate service to the user will occur based on these contexts and behaviours. Thus, adaptation required for any service will be initiated by the server. This is a strict method of enforcement since all controls and adaptations are carried out centrally on the server where CA-UCON RM is implemented.

The centralized architecture presents a number of benefits. Firstly, avoidance of duplication. A centralised approach facilitates having a single version of any information system for the entire organisation. They also help to ensure that every piece of data is stored only once. Secondly, sharing resources, since centralised system holds the information utilized across the organisation in single place, allowing access to it for all staff. This makes the undertaking of organisation-wide activities more efficient and straightforward.

Central process and planning also permits well-suited technology and skills to be established. The sharing of resources other than data is simplified. The transferral between units of hardware, software and employees becomes easier. Finally, scale economies are attained, since centralised approaches permit most activities to be carried out with lower unit cost.

From amongst the disadvantages of the centralized architecture is heavy time consumption, since carrying out actions and decisions centrally is additional time-consuming than non- centralized approach. This is due to the extra time it takes to assemble information from various different distributed places as input to centralised system decisions. Furthermore, when the central computer or database system fails,

then the system is inaccessible to anyone until the server/database recovers. Additionally, rigidity and increased dependence and susceptibility are caused by the centralized architecture.

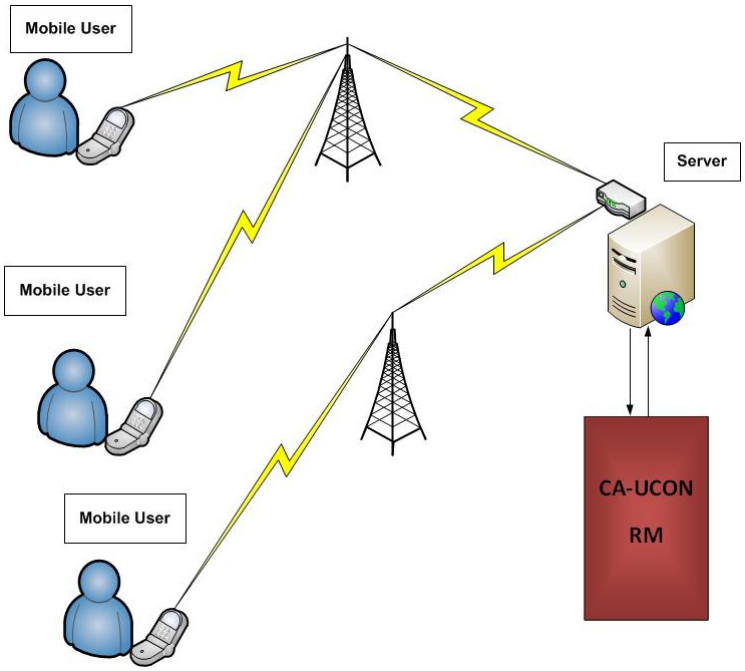


Figure 6.2: Centralised Enforcement Architecture

The following example demonstrates how the CA-UCON RM has been enforced in centralised architecture. We base our example on the case study mentioned in the previous chapter.

6.3.1.1 Example 1:

Suppose a ubiquitous learning (u-learning) system that provides u-lectures in three different formats: video, audio and text. The access to each format is restricted by specific requirement on context of the user. If the user requests a u-lecture in the video format when she/he is driving, the system will not deliver it to her/him in

the video format, but it will deliver it in the audio format which is suitable to the user current context (driving). In the centralised approach, the server may use a location tracking technique to sense the context of the user (whether driving or not) and then decide in what format the u-lecture must be delivered.

6.3.2 Distributed Enforcement Architecture

The distributed architecture can be understood in two distinct ways. It can be defined by the physical components or defined from the angle of the user or computation. These are known as the physical view and the logical view respectively. A distributed system is a set of nodes (computers or portable devices) connected by a communication network. The nodes in the network do not share their memory and are loosely coupled. The nodes in the system communicate via passing messages over the communication network using communication protocols.

The logical model is the view that an application has of the system. It includes a set of simultaneous processes and communication routes between them. The centre network is treated as completely linked. Communication of processes is done by transferring messages to each other. The CA-UCON RM will be enforced in the Mobile devices that will be responsible for observation of the context of the subject, object and environment. It can control the service or adapt to a new situation based on these contexts or behaviours as can be seen in Figure 6.3. In other words, the distributed architecture admits the implementing of the CA-UCON RM without a central unit to control and adapt the services.

The enforcement of distributed architecture exhibits a number of benefits. Prominent amongst these is greater compatibility between systems and local needs, since

users can develop their own information systems. These are more likely to correspond with their needs than those developed by someone else. Another benefit is greater system usage, as users show increased motivation by such approaches and are thus more enthusiastic to adopt computing when it directly bolsters their own interests, benefits and work. Finally, it supports quicker system development, since the less the organisational distance between system user and system developer, the swifter the development of that system is likely to be.

On the other hand certain negatives are displayed. This approach places barriers to sharing data. Distributed approaches can produce information systems in individual work units that are incompatible with each other and asynchronous. Further, effort is duplicated, since units will often duplicate what others are doing. Therefore distributed approaches tend to be very costly. Distribution can also lead to a failure to achieve scale economies as activities are not pooled.

The example below illustrates how the CA-UCON RM has been enforced in distributed architecture, which means that the control of services and adaptation process will be done on the client side. We use the following example from the case study mentioned in the previous chapter.

6.3.2.1 Example 2:

Following up from Example 1, suppose each u-lecture format requires some minimum amount of free memory available on the user's mobile device, e.g.: 5MB for a video, 2MB for an audio, and 1MB for a text format. If a user requests a u-lecture in video format and the available memory on her/his mobile device is less than 5MB,

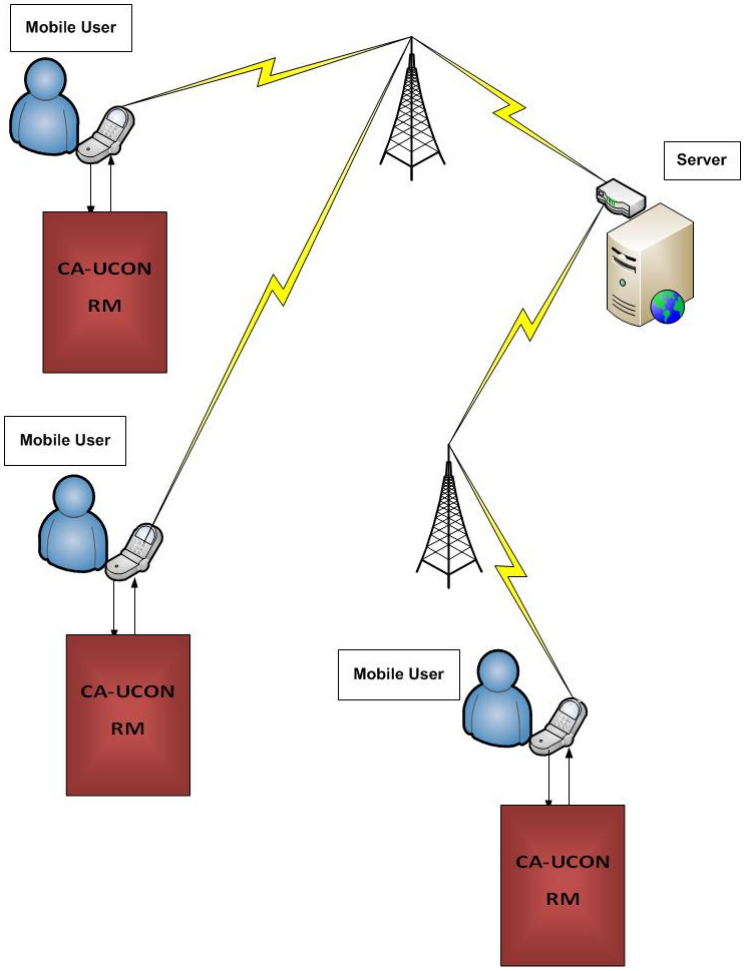


Figure 6.3: Distributed Enforcement Architecture

the system needs to adapt to this new situation. In a distributed enforcement architecture where CA-UCON RM is installed on each mobile device, the adaptation can be done e.g. by turning on a garbage collector software on the mobile device (client side). This will eventually increase the size of available memory on the mobile device. If enough memory has been freed and the current available memory on the mobile device is greater than 5MB, then the adaptation is successful and access will be granted.

6.3.3 Hybrid Enforcement Architecture

Hybrid enforcement architecture is an interesting evolutionary architecture that has combined features of centralised enforcement architecture and distributed enforcement architecture. The purpose is to circumvent the disadvantages displayed by these architectures whilst maintaining many of their advantages. In this architecture the CA-UCON RM is enforced in both the client side and the server side as can be seen in Figure 6.4.

Evaluation of the requested access right will take place on the mobile device side or on the server side, or on both. This enables the controlling of service or adapting to a new situation to occur at any time anywhere without restrictions. Some requested service accesses cannot be undertaken without adaptations from both the server and client side. This is difficult to do in the previously discussed architectures. On the other hand, the hybrid approach facilitates the implementation of the CA-UCON policies in the real world on both server and client sides, enabling all possible adaptations.

Hybrid architecture is a comprehensive and versatile approach. By combining two different architectures, a third, more tailored architecture for the enforcement of the CA-UCON RM has been evolved. Mobile devices and servers should be fitted with a variety of sensors in order to monitor all contexts related to the subject, object and environment. This is done so as to adapt to new situations and deliver a suitable service for the majority of contexts. For example, if the user requests a service in a certain context, but this service cannot be delivered in the given context based on CA-UCON RM, the server can communicate with the mobile device requiring it to adapt in order to receive a modified service.

The advantages of the hybrid architecture include reliability, because a fault detected in one part of the system can be isolated from the rest. Therefore the necessary corrective measures can be carried out, without disturbing the operating of any other part of the system. Furthermore, hybrid architecture displays more flexibility than other architectures since it integrates the benefits of both centralized and distributed architectures optimizing the existing resources whilst avoiding many drawbacks. On the other hand, the hybrid architecture involves a larger scale of work and therefore is normally more costly.

The subsequent instance shows how the CA-UCON RM has been enforced in the hybrid architecture. It can be seen that the control of services and adaptation process will be done on both sides, on the server side and on the client side. We present the following example based on the case study mentioned in the previous chapter.

6.3.3.1 Example 3:

In the u-learning system described in Example 1 and Example 2 above, if the user requests a u-lecture in video format when she/he is driving and the available memory size in her/his mobile device is less than 2MB, the system in this case needs to adapt to the new situation by performing actions on both sides (server side and client side). So, in the server side the system has to deliver the audio format instead of video format based on the user context which is in this case, driving. Meanwhile, the system has to check the amount of free memory available on the mobile device prior to delivering the service, which in this case is less than 2MB, i.e. lower than

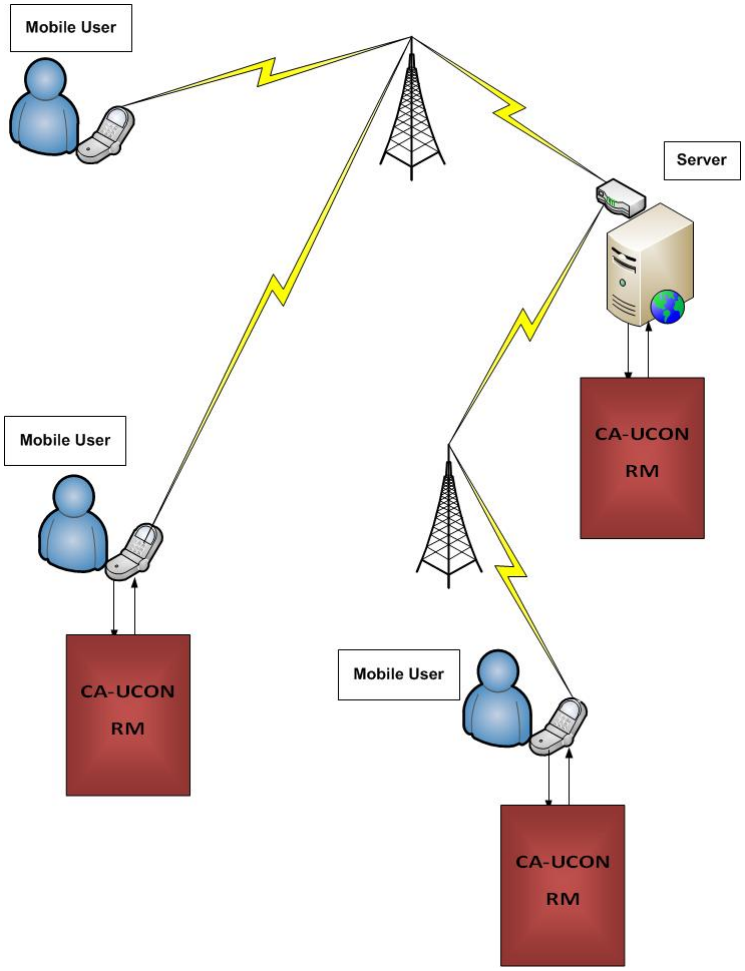


Figure 6.4: Hybrid Enforcement Architecture

the minimum memory size requirement for accessing a u-lecture in audio format. In this case the client side (mobile device) has to adapt as well, e.g. by executing a garbage collector software in order to free more memory space. If both adaptation actions are successful, then access will be granted; otherwise access will be denied. In this scenario, CA-UCON RM is installed on both the server and the user mobile device (client).

6.4 Summary

This chapter has investigated the enforcement architectures of CA-UCON model. We have introduced the architecture of CA-UCON reference monitor and explained its components. Moreover, we have presented three different enforcement architectures, discussing their advantages and disadvantages and the role of the CA-UCON RM within each. Finally, we presented an example for each type of enforcement architecture.

Chapter 7

Conclusions and Future Work

Objectives:

- Summaries the work in this thesis.
- Give the statement of evaluation
- List the main contributions of this research.
- Present future work that follows on from this thesis.

7.1 Work Summary

Producing an adaptive usage control model that is suited a dynamic environment, such as the Ubicomp environment, has been the ultimate aim of our work. This is needed because of the new security challenges that have been introduced by the Ubicomp environment, where information can be accessed and shared by users anytime, anywhere. In a Ubicomp system, the context of the environment is considered in order to control the service being requested and to adapt to any new situation. Producing such an adaptive usage control model has been a challenging task, due to the rapidly changing context of the Ubicomp environment, but the need for such a model to address the demands of a dynamic environment and to overcome the limitations of existing models has now been satisfied.

In meeting this goal, in this thesis we have proposed a context-aware and adaptive usage control model (CA-UCON) that extends the traditional usage control model (UCON) to enable adaptation to environmental changes with the aim of preserving continuity of usage in a pervasive computing system. In this model, we integrated the adaptation decision with the usage decision to generate a unified usage control model that is context-aware. The adaptation decision considers different types of contexts, such as the context of the subject, the object, the environment and the ICT before and during the access. Moreover, the formal definitions of the two newly introduced adaptation models within CA-UCON were presented: pre-adaptation and on-adaptation.

We then proposed a computational model for CA-UCON in order to demonstrate

how an access request on a resource is handled. This computational model was described formally as a Finite State Machine (FSM), which gives a comprehensive explanation of all the states and actions, in particular how and when the adaptation takes place (both before and during the access). The development of CA-UCON as an extension of UCON can be clearly seen from the FSM.

CCA was deemed the most appropriate formal notation for modelling and analysing the CA-UCON model; it is a mathematical notation used for modelling a system that is mobile and context-aware. We formally modelled CA-UCON using CCA and analysed its properties via the execution environment of CCA for validation purposes.

In order to assist in understanding the CA-UCON model and to evaluate its performance, a real-world case study of a ubiquitous learning (u-learning) system was presented. A modelling of this u-learning system in CA-UCON was given and we formally specified the u-learning system using the CCA mathematical notations. We then designed a number of scenarios and ran them using the CAA execution environment in order to analyse certain properties within our CA-UCON model.

We then illustrated the enforcement architecture of the CA-UCON model in a real situation. Firstly, we described a suitable reference monitor architecture and described in detail its components as well as their behaviour. We then presented the three types of enforcement architectures for CA-UCON and discussed their relative advantages and disadvantages in ubiquitous environments. Finally, we used a num-

ber of examples within ubiquitous environments in order to show in which situation each architecture can be most appropriately enforced.

7.2 Statement of Evaluation

Several works have been done in the area of access control in order to remedy their limitation in Ubicomp environment. Majority of these works have been conducted in traditional access control models such as (RBAC) [26][62]. A few of these researches have been accomplished in Usage control (UCON) model which is considered as latest enhancement to traditional access control models [113][39]. However, none of these extensions has wholly solved the limitation of access control models in Ubicomp environment. Unlike the previous extensions, CA-UCON model enables adaptation to environmental changes in the aim of preserving continuity of access by triggering specific actions to adapt to new situations and its main innovative feature is the integration of continuity of usage decision and dynamic adaptation to changes in the environmental or system context, so as to ensure continuity of usage. In addition to data protection, CA-UCON model enhances the quality of services, striving to keep explicit interactions with the user at a minimum.

In term of the formal specification used in this thesis, there are many existing formal specification which support mobility and/or context-aware, but majority of them were not appropriated in order to model context-aware mobile applications. For instance, one of these formal specification is called CONAWA proposed by [50] as calculus for applications that are context-aware, and this formal specification is inspired by calculus. The CONAWA's syntax concentrates on constructs that make it probable to navigate and describe via context. Another formal specification was

proposed by [65] which is known as Bigraphs, it is a unifying framework that used to model concurrent mobile systems, but it does not support context-awareness. Moreover, a UML specification of the infostation-based mLearning system was proposed by [33]. Despite the fact that UML specification offers a number of benefits like system analyses utilizing assistant tools and code generation, it has suffered from the need of formal reasoning support which represent its limitation for the design of critical systems. As consequence, we select CCA in order to model a context-aware system, as it is a mathematical notation that support mobility and context-awareness, as well as treats those primitive constructs as first-class citizens. However, to the best of our knowledge, none of these formal specification or others are used to model CA-UCON model.

Furthermore, regarding the case study that has been selected in this thesis, it is provably correct. The case study is modelled rely on our CA-UCON model and specified it utilizing the mathematical notation of CCA, then used the execution environment of ccaPL in order to validate some property-based scenarios.

7.3 Success Criteria Revisited

To answer the research questions that we pointed out in Chapter 1, a Context-Aware and Adaptive Usage Control model (CA-UCON in short) have been built and evaluated throughout the thesis. The objective of this work was to integrated the adaptation decision with usage decision in CA-UCON model to achieve a better performance that suits the Ubicomp environment. The analysis of our proposed model is presented using *CCA* and the validation of CA-UCON model properties is illustrated via the execution of the devised scenarios.

7.4 Contribution to Knowledge

The major contributions of the research in this thesis can be summarised as follows:

1. A context-aware and adaptive usage control model (CA-UCON) is developed, which is an extension of the UCON model, in order to function in a highly dynamic environment (ubiquitous environment) and to overcome the shortcoming of all previously proposed models.

2. An architecture of CA-UCON model is illustrated and comprehensive definition of its components are presented.

3. A computational model of CA-UCON is presented in order to demonstrate the behaviour of the various states when an access request is submitted.

4. The formal specification of CA-UCON is presented using the mathematical notations of CCA.

5. Formalising and analysing a real-world case study of ubiquitous learning (u-learning) system are successfully undertaken.

6. Analyzing some properties of CA-UCON, via proposing some scenarios in the u-learning environment and executing them through the execution environment of CCA, is achieved.

7. The possible enforcement architectures of the CA-UCON model are investigated and their advantages and disadvantages are presented.

7.5 Future Work

The Usage Control (UCON) model is still an active research area in the field of security; many issues remain to be considered, particularly with regard to the ubiquitous

environment. According to this research done in this thesis, the first route for future work is to extend our model to investigate further the enforcement mechanisms in order to derive new techniques and algorithms. This would require extensive work on security enforcement mechanisms in general and examining the existing proposed enforcement mechanisms in a dynamic environment in particular.

Another avenue for future work is the real implementation of a u-learning system using our CA-UCON approach for the purposes of further evaluation, where the dynamic (context-aware) environment would require an adaptive usage control in order to protect the resources and enhance the quality of services.

In terms of the extent of CCA, an additional route for future work is improving the ccaPL execution environment animator in order to enhance its layout and to enable any entity in the system to be visualised. This could be done by providing highly developed animation for those entities and improving additional aspects in the animation environment. In addition, there is a need for the model checker to be developed and added to the execution environment; this is important for validating any required specification of a system model. On the other hand, the CCA notation is still being developing and several notations are still under construction; these could be included in the syntax (for example, probability and time).

Bibliography

[1] H. Abie, P. Spilling, and B. Foyn. A distributed digital rights management model for secure information-distribution systems. *International Journal of Information Security*, 3:113–128, 2004.

[2] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, HUC '99, pages 304–307, London, UK, UK, 1999. Springer-Verlag.

[3] M. Alam, J.-P. Seifert, Q. Li, and X. Zhang. Usage control platformization via trustworthy selinux. In *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, ASIACCS '08, New York, NY, USA, 2008. ACM.

[4] D. Ardagna and B. Pernici. Adaptive service composition in flexible processes. *Software Engineering, IEEE Transactions on*, 33(6):369–384, 2007.

[5] G. Bai, L. Gu, T. Feng, Y. Guo, and X. Chen. Context-aware usage control for android. In *Security and Privacy in Communication Networks*, volume 50, pages 326–343. Springer Berlin Heidelberg, 2010.

[6] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context aware systems. *Int. J. Ad Hoc Ubiquitous Comput.*, 2(4):263–277, June 2007.

[7] T. Batista, A. Joolia, and G. Coulson. Managing dynamic reconfiguration in component-based systems. In *Proceedings of the 2nd European conference on Software Architecture*, EWSA'05, pages 1–17, Berlin, Heidelberg, 2005. Springer-Verlag.

[8] M. Beigl, A. Krohn, T. Zimmer, and C. Decker. Typical sensors needed in ubiquitous and pervasive computing. In *in Proceedings of the First International Workshop on Networked Sensing Systems (INSS & apos;04*, pages 153–158, 2004.

[9] P. Bellavista, A. Corradi, R. Montanari, and C. Stefanelli. Context-aware middleware for resource management in the wireless internet. *Software Engineering, IEEE Transactions on*, 29(12):1086–1099, 2003.

[10] E. Bertino, P. A. Bonatti, and E. Ferrari. Trbac: A temporal role-based access control model. *ACM Trans. Inf. Syst. Secur.*, 4(3):191–233, Aug. 2001.

[11] C. Bettini, D. Maggiorini, and D. Riboni. Distributed context monitoring for the adaptation of continuous services. *World Wide Web*, 10(4):503–528, Dec. 2007.

[12] M. Bishop. *Computer Security, Art and Science*. Addison-Wesley, 2003.

[13] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pages 164–173, 1996.

[14] C. Bolchini, C. A. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca. A data-oriented survey of context models. *SIGMOD Rec.*, 36(4):19–26, Dec. 2007.

[15] N. Boustia and A. Mokhtari. A dynamic access control model. *Applied Intelligence*, 36(1):190–207, Jan. 2012.

[16] P. Brown. Triggering information by context. *Personal Technologies*, 2:18–27, 1998.

[17] P. Brown, J. Bovey, and X. Chen. Context-aware applications: from the laboratory to the marketplace. *Personal Communications, IEEE*, 4(5):58–64, 1997.

[18] P. J. Brown. The Stick-e Document: a Framework for Creating Context-aware Applications. In *Proceedings of EP'96, Palo Alto*, pages 259–272. also published in it EP–odd, June 1996.

[19] L. Capra. Mobile computing middleware for context-aware applications. In *Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on*, pages 723–724, 2002.

[20] S.-H. Chang, H. J. La, and S. D. Kim. A comprehensive approach to service adaptation. In *Service-Oriented Computing and Applications, 2007. SOCA '07. IEEE International Conference on*, pages 191–198, 2007.

[21] H. Chen, T. Finin, and A. Joshi. Using owl in a pervasive computing broker, 2003.

[22] L. Chen and J. Crampton. On spatio-temporal constraints and inheritance in role-based access control. In *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, ASIACCS '08, pages 205–216, New York, NY, USA, 2008. ACM.

[23] A. Corrad, R. Montanari, and D. Tibaldi. Context-based access control management in ubiquitous environments. In *Network Computing and Applica-*

tions, 2004. (NCA 2004). Proceedings. Third IEEE International Symposium on, pages 253–260, 2004.

[24] A. K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7, Jan. 2001.

[25] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli. A survey of autonomic communications. *ACM Trans. Auton. Adapt. Syst.*, 1(2):223–259, Dec. 2006.

[26] C. dong Wang, T. Li, and L.-C. Feng. Context-aware environment-role-based access control model for web services. In *Multimedia and Ubiquitous Engineering, 2008. MUE 2008. International Conference on*, pages 288–293, 2008.

[27] C. Efstratiou, A. Friday, N. Davies, and K. Cheverst. Utilising the event calculus for policy driven adaptation on mobile systems. In *Policies for Distributed Systems and Networks, 2002. Proceedings. Third International Workshop on*, pages 13–24, 2002.

[28] Y. Fan, Z. Han, J. Liu, and Y. Zhao. A mandatory access control model with enhanced flexibility. In *Proceedings of the 2009 International Conference on Multimedia Information Networking and Security - Volume 01*, MINES '09, pages 120–124, Washington, DC, USA, 2009. IEEE Computer Society.

[29] J. B. Filho and H. Martin. A generalized context-based access control model for pervasive environments. In *Proceedings of the 2nd SIGSPATIAL ACM GIS 2009 International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '09, pages 12–21, New York, NY, USA, 2009. ACM.

[30] J. Fox and S. Clarke. Exploring approaches to dynamic adaptation. In *Proceedings of the 3rd International DiscCoTec Workshop on Middleware-Application Interaction*, MAI '09, pages 19–24, New York, NY, USA, 2009. ACM.

[31] M. Friedewald and O. Raabe. Ubiquitous computing: An overview of technology impacts. *Telematics and Informatics*, 28(2):55 – 65, 2011.

[32] I. Ganchev and M. O'Droma. Mobile distributed e-learning center. In *Proceedings of the Fifth IEEE International Conference on Advanced Learning Technologies*, ICALT '05, pages 593–594, Washington, DC, USA, 2005. IEEE Computer Society.

[33] I. Ganchev, S. Stojanov, M. ODroma, and D. Meere. An infostation-based multi-agent system supporting intelligent mobile services across a university campus. *Journal of Computers*, 2(3), 2007.

[34] S. Haibo and H. Fan. A context-aware role-based access control model for web services. In *e-Business Engineering, 2005. ICEBE 2005. IEEE International Conference on*, pages 220–223, 2005.

[35] R. Hamadi and B. Benatallah. Recovery nets: Towards self-adaptive workfow systems. In *in Proceedings of the 5th International Conference on Web Information Systems Engineering (WISE 04), LNCS 3306*, pages 439–453. Springer Verlag, 2004.

[36] M. A. Hiltunen and R. D. Schlichting. Adaptive distributed and fault-tolerant systems. *International Journal of Computer Systems Science and Engineering*, 11:125–133, 1995.

[37] M. Hilty, A. Pretschner, D. Basin, C. Schaefer, and T. Walter. A policy language for distributed usage control. In *Proceedings of the 12th European conference on Research in Computer Security*, ESORICS'07, pages 531–546, Berlin, Heidelberg, 2007. Springer-Verlag.

[38] M. Hilty, A. Pretschner, C. Schaefer, and T. Walter. Usage control requirements in mobile and ubiquitous computing applications. In *Proceedings of the*

International Conference on Systems and Networks Communication, ICSNC '06, pages 27–, Washington, DC, USA, 2006. IEEE Computer Society.

[39] Z. Hong-jun. *Study and application of special access control model based on UCON*. PhD thesis, Nanjing: Jiangshu University, May 2009.

[40] H. Janicke, A. Cau, and H. Zedan. A note on the formalisation of ucon. In *Proceedings of the 12th ACM symposium on Access control models and technologies*, SACMAT '07, pages 163–168, New York, NY, USA, 2007. ACM.

[41] J. Joshi, E. Bertino, and A. Ghafoor. Hybrid role hierarchy for generalized temporal role based access control model. In *Computer Software and Applications Conference, 2002. COMPSAC 2002. Proceedings. 26th Annual International*, pages 951–956, 2002.

[42] E. B. W. Jr. *An Introduction to Scientific Research*. Dover Publications, 1991.

[43] K. Kakousis, N. Paspallis, and G. A. Papadopoulos. A survey of software adaptation in mobile and ubiquitous computing. *Enterp. Inf. Syst.*, 4(4):355–389, Nov. 2010.

[44] V. Kapsalis, L. Hadellis, D. Karelis, and S. Koubias. A dynamic context-aware access control architecture for e-services. *Computers & Security*, 25:507 – 521, 2006.

[45] A. Kayes, J. Han, and A. Colman. Icaf: A context-aware framework for access control. In *Information Security and Privacy*, volume 7372, pages 442–449. Springer Berlin Heidelberg, 2012.

[46] J. Kephart and R. Das. Achieving self-management via utility functions. *Internet Computing, IEEE*, 11(1):40–48, 2007.

[47] M. Khan and K. Sakamura. Context-aware access control for clinical information systems. In *Innovations in Information Technology (IIT), 2012 International Conference on*, pages 123–128, 2012.

[48] K. 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*, ICNICONSMCL '06, pages 123–, Washington, DC, USA, 2006. IEEE Computer Society.

[49] M. Kirkpatrick and E. Bertino. Context-dependent authentication and access control. In *iNetSec 2009 – Open Research Problems in Network Security*, volume 309, pages 63–75. Springer Berlin Heidelberg, 2009.

[50] M. B. Kjrgaard, J. Bunde-pedersen, C. C, M. B. Kjrgaard, M. Baun, and K. J. Bunde-pedersen. This document in subdirectoryrs/06/2/ conawa: A formal model for context awareness, 2006.

[51] P. Korpipaa, J. Mantyjarvi, J. Kela, H. Keranen, and E.-J. Malm. Managing context information in mobile devices. *Pervasive Computing, IEEE*, 2(3):42–51, 2003.

[52] W. Ku and C. hung Chi. Survey on the technological aspects of digital rights management. In *In ISC*, pages 391–403, 2004.

[53] A. Lazouski, F. Martinelli, and P. Mori. Survey: Usage control in computer security: A survey. *Comput. Sci. Rev.*, 4(2):81–99, May 2010.

[54] P. Lefrere. Activity-based scenarios for and approaches to ubiquitous e-learning. *Personal Ubiquitous Comput.*, 13(3):219–227, Mar. 2009.

[55] N. Li. How to make discretionary access control secure against trojan horses. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–3, 2008.

[56] Q. Liu, R. Safavi-Naini, and N. P. Sheppard. Digital rights management for content distribution. In *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003 - Volume 21*, ACSW Frontiers '03, pages 49–58, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[57] S. LOKE. *Context-Aware Pervasive system: Architectures for new breed of applications*. Auerbach Publications, Wiley, 2006.

[58] M. A. Luo Xiaofeng, Li Ling and L. Wanbo. The contextual usage control model. *Zhejiang University Science (Computers & Electronics)*, 2012.

[59] K. Mandula, S. Meda, D. Jain, and R. Kambham. Implementation of ubiquitous learning system using sensor technologies. In *Technology for Education (T4E), 2011 IEEE International Conference on*, pages 142–148, 2011.

[60] F. Martinelli, P. Mori, and A. Vaccarelli. Towards continuous usage control on grid computational services. In *Proceedings of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services*, ICAS-ICNS '05, pages 82–, Washington, DC, USA, 2005. IEEE Computer Society.

[61] M. Matson and M. Ulieru. The 'how' and 'why' of persistent information security. In *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services*, PST '06, New York, NY, USA, 2006. ACM.

[62] M. C. Matthew, . Matthew, J. Moyer, and M. Ahamad. Generalized role-based access control for securing future applications, 2000.

[63] P. K. Mckinley, S. M. Sadjadi, E. P. Kasten, and B. H. C. Cheng. A taxonomy of compositional adaptation. Technical report, Department of Compyter Science and Engineering, Michigan State University, 2004.

[64] R. Milner. *Communicating and Mobile Systems: The Pi Calculus*. Cambridge University Press, 1999.

[65] R. Milner. Pure bigraphs: structure and dynamics. *Inf. Comput.*, 204(1):60–122, Jan. 2006.

[66] M. Nauman and T. Ali. Hue: A hardware ucon engine for fine-grained continuous usage control. In *Multitopic Conference, 2008. INMIC 2008. IEEE International*, pages 59–64, 2008.

[67] R. Nick, J. Pascoe, and D. Morse. Enhanced reality fieldwork: the context-aware archaeologist assistant. In Exon, editor, *Computer Applications & Quantitative Methods in Archaeology*, volume 0. Archaeopress, 1997.

[68] S. Oh and S. Park. Task-role-based access control model. *Inf. Syst.*, 28(6):533–562, Sept. 2003.

[69] 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*, MUE '07, pages 1158–1163, Washington, DC, USA, 2007. IEEE Computer Society.

[70] P. Oreizy, N. Medvidovic, and R. N. Taylor. Runtime software adaptation: framework, approaches, and styles. In *Companion of the 30th international*

conference on Software engineering, ICSE Companion '08, New York, NY, USA, 2008. ACM.

[71] S. Owicki and L. Lamport. Proving liveness properties of concurrent programs. *ACM Trans. Program. Lang. Syst.*, 4(3):455–495, July 1982.

[72] J. Park and R. Sandhu. Towards usage control models: beyond traditional access control. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, SACMAT '02, pages 57–64, New York, NY, USA, 2002. ACM.

[73] J. Park and R. Sandhu. The ucon $_{ABC}$ usage control model. *ACM Trans. Inf. Syst. Secur.*, 7(1):128–174, Feb. 2004.

[74] J. Park, X. Zhang, and R. S. Attribute mutability in usage control. In *In Proceedings of the Proceedings of 18th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, pages 15–29. Kluwer, 2004.

[75] M. J. Pascoe. Adding generic contextual capabilities to wearable computers. In *Proceedings of the 2nd IEEE International Symposium on Wearable Computers*, ISWC '98, pages 92–, Washington, DC, USA, 1998. IEEE Computer Society.

[76] C. Piao and X. Gan. Research on trust management model for e-commerce based on fuzzy clustering method. In *e-Business Engineering, 2009. ICEBE '09. IEEE International Conference on*, pages 188–195, 2009.

[77] S. Poslad. *Ubiquitous Computing: Smart Devices, Environments and Interactions*. Auerbach Publications, Wiley, 2009.

[78] A. Pretschner, M. Hilty, and D. Basin. Distributed usage control. *Commun. ACM*, 49(9):39–44, Sept. 2006.

[79] A. Pretschner, F. Massacci, and M. Hilty. Usage control in service-oriented architectures. In C. Lambrinoudakis, G. Pernul, and A. Tjoa, editors, *Trust, Privacy and Security in Digital Business*, volume 4657 of *Lecture Notes in Computer Science*, pages 83–93. Springer Berlin Heidelberg, 2007.

[80] D. Salber, A. K. Dey, and G. D. Abowd. Ubiquitous computing: Defining an hci research - agenda for an emerging interaction paradigm. Technical report, 1998.

[81] D. Salber, A. K. Dey, and G. D. Abowd. The context toolkit: aiding the development of context-enabled applications. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, CHI '99, pages 434–441, New York, NY, USA, 1999. ACM.

[82] M. Salehie and L. Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Trans. Auton. Adapt. Syst.*, 4(2):14:1–14:42, May 2009.

[83] P. Samarati and S. D. C. d. Vimercati. Access control: Policies, models, and mechanisms. In *Revised versions of lectures given during the IFIP WG 1.7 International School on Foundations of Security Analysis and Design on Foundations of Security Analysis and Design: Tutorial Lectures*, FOSAD '00, pages 137–196, London, UK, UK, 2001. Springer-Verlag.

[84] P. Samarati and S. D. C. d. Vimercati. Access control: Policies, models, and mechanisms. In *Revised versions of lectures given during the IFIP WG 1.7 International School on Foundations of Security Analysis and Design on Foundations of Security Analysis and Design: Tutorial Lectures*, FOSAD '00, London, UK, UK, 2001. Springer-Verlag.

[85] R. Sandhu and J. Park. Usage control: A vision for next generation access control. In V. Gorodetsky, L. Popyack, and V. Skormin, editors, *Computer*

Network Security, volume 2776 of *Lecture Notes in Computer Science*, pages 17–31. Springer Berlin Heidelberg, 2003.

[86] R. Sandhu and P. Samarati. Access control: principle and practice. *Communications Magazine, IEEE*, 32(9):40–48, 1994.

[87] R. Sandhu, X. Zhang, K. Ranganathan, and M. J. Covington. Client-side access control enforcement using trusted computing and pei models. *Journal of High Speed Networks*, 15(3), May 2010.

[88] R. S. Sandhu. Lattice-based access control models. *Computer*, 26(11):9–19, Nov. 1993.

[89] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *Computer*, 29(2):38–47, Feb. 1996.

[90] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *Computer*, 29(2):38–47, Feb. 1996.

[91] D. Sangiorgi and D. Walker. *PI-Calculus: A Theory of Mobile Processes*. Cambridge University Press, New York, NY, USA, 2001.

[92] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, WMCSA '94, pages 85–90, Washington, DC, USA, 1994. IEEE Computer Society.

[93] B. Schilit and M. Theimer. Disseminating active map information to mobile hosts. *Network, IEEE*, 8(5):22–32, 1994.

[94] P. Schneck. Persistent access control to prevent piracy of digital information. *Proceedings of the IEEE*, 87(7):1239–1250, 1999.

[95] F. Siewe, A. Cau, and H. Zedan. A compositional framework for access control policies enforcement. In *Proceedings of the 2003 ACM workshop on Formal methods in security engineering*, FMSE '03, pages 32–42, New York, NY, USA, 2003. ACM.

[96] F. Siewe, H. Zedan, and A. Cau. The calculus of context-aware ambients. *J. Comput. Syst. Sci.*, 77(4):597–620, July 2011.

[97] J. Solworth and R. Sloan. A layered design of discretionary access controls with decidable safety properties. In *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, pages 56–67, 2004.

[98] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*, 2004.

[99] K. Thi, T. Dang, P. Kuonen, and H. Drissi. Strobac: Spatial temporal role based access control. In *Computational Collective Intelligence. Technologies and Applications*, volume 7654 of *Lecture Notes in Computer Science*, pages 201–211. Springer Berlin Heidelberg, 2012.

[100] M. Toahchoodee, I. Ray, K. Anastasakis, G. Georg, and B. Bordbar. Ensuring spatio-temporal access control for real-world applications. In *Proceedings of the 14th ACM symposium on Access control models and technologies*, pages 13–22, New York, NY, USA, 2009. ACM.

[101] E. Uzun, V. Atluri, S. Sural, J. Vaidya, G. Parlato, A. L. Ferrara, and M. Parthasarathy. Analyzing temporal role based access control models. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, SACMAT '12, pages 177–186, New York, NY, USA, 2012. ACM.

[102] H. Wang, Y. Zhang, and J. Cao. Access control management for ubiquitous computing. *Future Generation Computer Systems*, 24(8):870 – 878, 2008.

[103] J. Wang and X. Fu. Digital rights management (drm) in the mobile p2p environment. In *Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on*, pages 1–4, 2010.

[104] S. Weeks. Understanding trust management systems. In *Security and Privacy, 2001. S P 2001. Proceedings. 2001 IEEE Symposium on*, pages 94–105, 2001.

[105] M. Weiser. The computer for the twenty-first century. In *Scientific American*, pages 265(3):94-104, 1991.

[106] D.-Z. Xu, D. Xu, and Z. Lei. Bigraphical model of context-aware in ubiquitous computing environments. *Asia-Pacific Conference on Services Computing. 2006 IEEE*, 0:389–394, 2011.

[107] S.-H. Yang and I. Chen. Providing context aware learning services to learners with portable devices. In *Advanced Learning Technologies, 2006. Sixth International Conference on*, pages 840–842, 2006.

[108] G. Zhang and M. Parashar. Dynamic context-aware access control for grid applications. In *Grid Computing, 2003. Proceedings. Fourth International Workshop on*, pages 101–108, 2003.

[109] H. Zhang, Y. He, and Z. Shi. Spatial context in role-based access control. In *Proceedings of the 9th international conference on Information Security and Cryptology*, ICISC'06, pages 166–178, Berlin, Heidelberg, 2006. Springer-Verlag.

[110] X. Zhang, F. Parisi-Presicce, R. Sandhu, and J. Park. Formal model and policy specification of usage control. *ACM Trans. Inf. Syst. Secur.*, 8(4):351–387, Nov. 2005.

[111] X. Zhang, J. Park, F. Parisi-Presicce, and R. Sandhu. A logical specification for usage control. In *Proceedings of the ninth ACM symposium on Access control models and technologies*, SACMAT '04, pages 1–10, New York, NY, USA, 2004. ACM.

[112] Z. Zhang, L. Yang, Q. Pei, and J. Ma. Research on usage control model with delegation characteristics based on om-am methodology. In *Network and Parallel Computing Workshops, 2007. NPC Workshops. IFIP International Conference on*, pages 238–243, 2007.

[113] B. Zhao, R. Sandhu, X. Zhang, and X. Qin. Towards a times-based usage control model. In *Proceedings of the 21st annual IFIP WG 11.3 working conference on Data and applications security*, pages 227–242, Berlin, Heidelberg, 2007. Springer-Verlag.

[114] X. Zhao, X. Wan, and T. Okamoto. Adaptive content delivery in ubiquitous learning environment. In *Wireless, Mobile and Ubiquitous Technologies in Education (WMUTE), 2010 6th IEEE International Conference on*, pages 19–26, 2010.

[115] J. Zheng, kun Zhang, wen Zheng, and an Tan. Dynamic role-based access control model. *Journal of Software*, 6(6), 2011.