View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Arrow@dit

Technological University Dublin

ARROW@TU Dublin

Doctoral

Science

2015

A Knowledge-driven Distributed Architecture for Context-Aware Systems

Dennis Lupiana

Technological University Dublin, dennis.lupiana@student.dit.ie

Follow this and additional works at: https://arrow.tudublin.ie/sciendoc

 Part of the Computer Engineering Commons

Recommended Citation

Lupiana, D. (2015) A Knowledge-driven Distributed Architecture for Context-Aware Systems, Doctoral Thesis, Technological University Dublin. doi:10.21427/D77C7T

This Theses, Ph.D is brought to you for free and open access by the Science at ARROW@TU Dublin. It has been accepted for inclusion in Doctoral by an authorized administrator of ARROW@TU Dublin. For more information, please contact yvonne.desmond@tudublin.ie, arrow.admin@tudublin.ie, brian.widdis@tudublin.ie.

This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 License

A Knowledge-driven Distributed Architecture for Context-Aware Systems

Dennis Lupiana, *MSc.*

Supervised by;
Dr. Fredrick J. Mtenzi, Mr. Ciaran O'Driscoll
and Prof. Brendan O'Shea

School of Computing

Dublin Institute of Technology, Ireland

Thesis submitted to the Office of Postgraduate Studies and Research at the Dublin Institute of Technology in the fulfilment of the requirements for the Degree of Doctor of Philosophy

January, 2015

To my family.

Abstract

As the number of devices increases, it becomes a challenge for the users to use them effectively. Interacting with these devices becomes difficult and more time consuming. This is more challenging when the majority of these devices are mobile. The users and their devices enter and leave different environments where different settings and computing needs may be required. To effectively use these devices in such environments means to constantly be aware of their whereabouts, functionalities and desirable working conditions. This is impractical and hence it is imperative to increase seamless interactions between the users and devices, and to make these devices less intrusive.

To address these problems, various *responsive* computing systems, called *context-aware* systems, have been developed. These systems rely on architectures to perceive their physical environments in order to appropriately and effortlessly respond. Currently, the majority of the existing architectures focus on acquiring data from sensors, interpreting and sharing it with these systems. These architectures are developed with a limited model of the real *world* in which the users and devices interact. Little has been done to develop a comprehensive model of the real world, and an architecture that can use available information to perceive and hence to enable these systems to adapt to social dynamics.

This research addresses these limitations by proposing a Knowledge-driven Distributed Architecture (KoDA). Centre to KoDA is a Knowledge-intensive Context Model (K*i*CM). K*i*CM enables knowledge about the *real world* to be represented in KoDA. This knowledge enables KoDA to use available information to dynamically perceive an environment and its dynamics. This research shows that the accuracy of situation recognition increases significantly when knowledge of the users and their computer-related activities, the users' devices, location and time is used. This research also shows that this accuracy is further increased when knowledge of certainty level of each sensor is also used.

Declaration

I certify that this thesis which I now submit for examination for the award of Doctor of Philosophy, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work.

This thesis was prepared according to the regulations for postgraduate study by research of the Dublin Institute of Technology, hereafter referred to as the Institute, and has not been submitted in whole or in part for another award in any Institute.

The work reported on in this thesis conforms to the principles and requirements of the Institute's guidelines for ethics in research.

The Institute has permission to keep, lend or copy this thesis in whole or in part, on condition that any such use of the material of the thesis be duly acknowledged.

. . . . . .

Dennis Lupiana

January, 2015

Acknowledgements

I would like to express my sincere gratitude and appreciation to my supervisors; Dr. Fredrick Mtenzi, Mr. Ciaran O'Driscoll and Prof. Brendan O'Shea for their invaluable time, effort and expertise. I could not have imagined having a better supervision team. Special thanks to Prof. Brendan who had to commute, approximately 40 kilometres, occasionally to meet with me. Special thanks also to Dr. Fredrick who has been more than a supervisor to me.

Many thanks to my examiners, Prof. Mike Wald and Dr. Susan McKeever, and the chairperson of my defence, Dr. Anthony Betts, for their insightful comments. I would also like to thank my government and the administration of the Institute of Finance Management (IFM) for allocating funds for this research. Many thanks also to the administration of the School of Computing, DIT for giving me the opportunities to participate in teaching activities.

I would also like to express my appreciation to my colleagues, staff and students from the School of Computing, DIT and my friends, Daisy, Flavio and Bhavin, for participating in my experiments. Many thanks also to my colleagues from room 3030 Aungier Street, 34 New Bride Street and A109 Kevin Street, DIT. They have always been supportive and encouraging.

Many thanks also to my parents, my brother and my sisters for being so supportive and encouraging. Special thanks to my beautiful and loving wife, Vallerie, and to my loving kids, Lunotso and A'bema. My wife has always been there for me, celebrating with me in any progress I made and encouraging me whenever I felt lost. With warm welcome from my kids whenever I come back home, and a series of funny games afterwards, every morning I felt like a new person.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Ericsson (2011) predicts that more than fifty billion devices will be connected by 2020. Evans (2011), from Cisco, estimates the number of devices connected per person to be more than three in 2015, and more than six in 2020. This figure will drastically increase if the number of *'anti-technology'* people is excluded. These estimations show the sheer number of devices, whether connected to the Internet or not, that will be at the users' disposal. The survey conducted by Sophos lab[1] indicates that currently one person is estimated to carry three mobile devices; a Smartphone, laptop and tablet. This number, as the trend shows, will continue to increase. By including stationary devices that the users interact with and devices embedded almost everywhere, this figure will drastically increase.

This trend will have many benefits on the way people accomplish their daily activities. While the future is promising, as the number of devices increases it becomes a challenge for the users to effectively use them. Interacting with devices becomes difficult and more time consuming. This is more challenging when the majority of the devices are mobile. The users and their devices enter and leave different environments where different settings and computing needs may be required. To effectively use devices in such environments means to constantly be aware of their whereabouts, functionalities, and desirable working conditions. This is impractical and hence it is imperative to increase seamless interactions between the users and devices, and to make these devices less intrusive.

To date, there has been a lot of effort devoted to develop context-aware systems

[1] http://www.sophos.com/en-us/press-office/press-releases/2013/03/mobile-security-survey.aspx

and their supporting architectures, known as *context-aware architectures*. These systems utilise information about entities, such as the users, in an environment to make devices respond appropriately (Schilit *et al.*, 1994). The architectures enable these systems to understand their environments and the users' computing needs. A typical application of a context-aware system is when a person's Smartphone automatically switches to a silence mode when the person attends a meeting. Such systems significantly reduce physical interactions between the users and devices, and make devices less intrusive. The question is, however, *How would a context-aware system know when people are chatting or having a meeting in order to make devices respond appropriately?*

This research addresses this question by proposing a knowledge-driven distributed context-aware architecture. This research is in the Context-Awareness research domain. This research domain seeks to realise the invisible nature of devices as envisaged in Ubiquitous Computing (UbiComp). The pioneers of UbiComp, Weiser and his colleagues (1991), envisaged a *world* where people interact and use hundreds of devices subconsciously. Hence, this research contributes to making the use of devices intuitive. The discussion of the implications of this vision and their research related mainstreams is provided in chapter 2. Section 1.2 provides definitions of the key concepts used in this thesis.

1.2 Key Definitions

There are various definitions of context (Chen, 2004; Coutaz *et al.*, 2005; Dey, 2000; Kofod-Petersen, 2007; McKeever, 2011; Schilit *et al.*, 1994, 2003; Schmidt

et al., 1999b; Zimmermann *et al.*, 2007). These definitions either refer to context as an attribute of an entity, which is essential for a context-aware system to accomplish its tasks, or as an abstraction of circumstances that influence a context-aware system on how it operates. To differentiate these interpretations, this thesis abstracts the latter and the former definitions of context with *context parameter* and *situation*, respectively. The rest of this section defines context parameter, situation and context model as used in this thesis.

1.2.1 A Context Parameter

In this thesis a context parameter is defined as a piece of meaningful information about an entity that has an impact on a context-aware system. This information may be interpreted from data captured by a sensor or acquired directly from other sources such as a network or an application software. In this thesis, for example, the name of the owner of a device interpreted from the device's ID captured by a sensor is a context parameter. The status of the user's keyboard or mouse usage is also a context parameter in this thesis. As referred by others, these parameters are collectively referred to as *low-level* contexts.

A widely used synonym of a context parameter is a *contextual information*. This term, however, is interchangeably used with singular and plural meaning. Gu *et al.* (2005) and Chen (2004), for instance, refer to identity, location or time as a contextual information while Ye *et al.* (2007) and Henricksen (2003) refer to a set of context parameters as contextual information. Hence, to avoid this confusion, the term context parameter is preferred in this thesis. In occasions that this thesis refers to a related work that use contextual information in plural

form, the term context parameters will be used.

1.2.2 A Situation

This research adopts the definition of a situation from Kofod-Petersen (2007) who defines a situation as a social setting, such as a meeting, where the users involved want to achieve various goals. This definition of situation differs from that of Dey (2000), Henricksen (2003), and Ranganathan and Campbell (2003a) as is not confined to a particular task. This definition emphasises meaningful interactions between relevant entities required to sufficiently describe the real world that is of interest to the users and their devices. A situation provides a detailed picture of the real world environment whereas a context parameter provides an aspect of the real world environment.

In situation, the process of reaching a decision to invoke context-aware systems is complex. This process involves using various relevant context parameters simultaneously rather than using one context parameter separately. In situation, for instance, a decision to switch users mobile phone into a silent mode is not simply reached when the user enters a meeting venue. This decision, for instance, is reached after taking into account information about the venue, existing users in the venue, their social relations and activities, status of their devices and time.

1.2.3 A Context Model

> *"To provide more interesting and useful applications...we must expect to tackle difficult issues of knowledge modelling and representation"*
> *Dey (2000)*

This research adopts the definition of a model from Gregory (1993) who defines a model as a simplified representation of a certain reality. The reality that this research is interested with is the users' situations. Hence, in this research a context model is an abstract representation of meaningful relationships between relevant entities required to sufficiently describe a situation. This definition of a context model is important as it advocates for a knowledge-intensive and generic context model, which is preferred in context-awareness (Dey, 2000; Strang & Linnhoff-Popien, 2004). The analysis and discussion of the existing context models is provided in section 3.3.

In this research, a context model supports a developer in different phases of implementation. In the design phase, a developer uses the model to identify sensing technologies required to monitor and capture different aspects of the real world. In the implementation phase, a developer uses the model to systematically identify relevant knowledge about situations and as a guideline for representing it in a context-aware architecture. Consequently, this knowledge facilitates a context-aware architecture to reason about information it collects from a physical environment and hence to recognise ongoing situation.

1.3 Research Motivation

As more devices emerge, it becomes difficult and more time consuming for the users to interact with and effectively use them. This is more challenging when the majority of the devices are mobile. The users and their devices enter and leave different environments where different settings and computing needs may be required. To effectively use the devices in such environments means to constantly be aware of their whereabouts, functionalities, and desirable working conditions. This is impractical and hence it is imperative to increase seamless interactions between the users and devices, and to make these devices less intrusive.

Researchers respond to these problems by developing *smart* artefacts (Kortuem *et al.*, 2010; Schmidt & Van Laerhoven, 2001; Streitz *et al.*, 2005) and context-aware systems (Kukkonen *et al.*, 2009; Liu, 2010; van de Westelaken *et al.*, 2011). However, developing algorithms to recognise the user's situations (Cimino *et al.*, 2012; Li *et al.*, 2013; McKeever, 2011), frameworks (Biegel, 2005; Dey, 2000; Henricksen, 2003), middleware (Da *et al.*, 2014; Ranganathan & Campbell, 2003b; Roalter *et al.*, 2010) and architectures (Chen, 2004; Kaenampornpan, 2009; Kofod-Petersen, 2007) to support context-aware systems remain long standing challenges. While each solution addresses the problem in its own unique way, designing context-aware architectures, which is the focus of this research, is crucial to these problems. To fully take advantage of these architectures, however, their designs should also take into account social dynamics.

1.4 Problem Statement

Research in Context-Awareness has proposed various architectures to support context-aware systems. The architectures also support the development of *active* computing environments called *Smart environments*. These architectures enable context-aware systems to respond to information about entities within their proximity. Subsequently, this enables Smart environments to automate repetitive tasks and to automatically provide user-tailored computing needs. The users, for instance, can enter a Smart meeting room without worrying about the settings of their mobile phones. Indeed, the research has significantly reduced physical interactions between the users and devices.

Currently, however, the majority of the existing architectures focus on acquiring data from sensors, interpreting it and sharing the resultant context parameters with context-aware systems. These architectures are designed with little or no consideration of social dynamics, or *situations*. These architectures are designed with a limited representation of the real *world* in which the users and devices interact. Hence, these architectures fail to exploit available information to dynamically recognise ongoing situations and hence to effectively and intelligently support context-aware systems. As a result, context-aware systems respond to individual context parameters rather than to occurring situations.

These architectures are designed based on limited context models. These models are limited to the representation of knowledge about; (M_1) a particular entity, focusing on its specific attribute (Schilit, 1995), (M_2) an entity, its subclasses and its attributes (Chen, 2004) or (M_3) different entities, relationships amongst them

and their attributes (Kaenampornpan, 2009; Kofod-Petersen, 2007). The initial models, M_1 and M_2, are developed for architectures that support context-aware systems that automate repetitive tasks and hence fail to specify relationships between different entities in an environment. The recent models, M_3, are developed for architectures that support context-aware systems that provide personalised computing needs and hence only few entities are considered to be relevant.

Therefore, the problem of how knowledge about different entities within a user's proximity can be effectively used to enable context-aware architectures to dynamically recognise ongoing situations is an open research problem. This research aims to address this problem. In a broader sense, this problem is twofold;

P_1 - Investigating *what* and *how* different entities within a physical environment can be used to develop a generic and comprehensive model of situations

P_2 - Investigating *how* a context-aware architecture can be designed to dynamically and accurately recognise ongoing situations

This is a difficult problem. The number of entities that can be used to represent a situation is numerous and relationships amongst them are dynamic. In addition, as the users and their devices become mobile, computing environments become *open* and hence *dynamic*. A slight change in such environments can have a huge impact on the use of the devices and the users' computing needs. To keep up with these changes is a big challenge. Questions such as what changes should be taken into account and how often, need to be addressed. Furthermore, the majority of devices are resource-constrained and thus, to acquire, interpret,

store, aggregate and reason about abundant information about relevant entities from an environment is a challenge.

1.5 Thesis Statement

This research hypothesises that if the relations between the users, an environment, time, computing devices and computing services are specified and utilised, then a context-aware architecture can dynamically and accurately recognise the users' ongoing situations.

1.6 Research Methodology

> *"Science classifies knowledge. Experimental science classifies knowledge derived from observations." Denning (1980)*

In this research, a Knowledge-driven Distributed Architecture (KoDA) and a generic and Knowledge-intensive Context Model (KiCM) are proposed. To experiment with KoDA, a prototype is implemented. As noted by Tichy (1998) and Weiser (1991), experiments are crucial in scientific research in Computer Science and are core to the evaluation of any UbiComp system. In line with Tichy (1998) and Weiser (1991), this research has conducted a set of experiments. Hence, this research is in line with the experimental computer science research (Denning, 1980).

To address P_1, relevant literature on *Actor-Network Theory* (Callon, 1991) and *Semantic Network* (Sowa, 1991; Woods, 1975) is reviewed. To identify the entities and relationships required to develop the model, the *framing* and *disentanglement* principles of the theory are used. The theoretical structure of the theory is adopted to represent the entities and their relationships. To conceptually represent the model, the Semantic Network representation formalisms are adopted. Chapter 4 provides a discussion of *why* the theory and the formalisms are preferred and *how* they are applied in this research.

To address P_2, the work by Chen (2004), Kofod-Petersen (2007), Kaenampornpan (2009) and other relevant literature is reviewed to design KoDA with the ability to (A_1) sense and interpret information about relevant entities in environments (A_2) represent and reason with knowledge about situations and information about the entities and (A_3) run required applications. A_1, A_2 and A_3 form a *perception layer*, *inference layer* and *application layer* of KoDA respectively. Dynamism and distributed nature are the key features of any potential solution and therefore KoDA is designed to incorporate them. To illustrate the capabilities of KoDA and to experiment with it, a prototype is implemented.

To implement the perception layer, a number of technologies have been used. A Radio Frequency Identification (RFID) technology has been used to identify the users and the room. This research also implements Keyboard Activity Monitor (KAM) and Mouse Activity Monitor (MAM) to remotely monitor the users' keyboard and mouse activities respectively. To monitor status of the users' computers and time, Java functions are also implemented. In this research a pair of an RFID reader and antenna is referred to as a *physical sensor* while KAM,

MAM and the Java functions are referred to as *logical sensors*. This research uses eXtensible Markup Language[1] (XML) and its Java API[2] to represent and interpret information about the entities respectively.

To implement the inference layer, both logical-based and probabilistic inference mechanisms have been used, separately. In particular, *Rete* algorithm (Forgy, 1979) and the Microsoft Bayesian Network API[3] (MSBN3) have been used to implement rule-based and Bayesian inference mechanisms respectively. Consequently, rule-based knowledge representation language and Bayesian network have been used to represent knowledge about situations in the prototype. The Bayesian inference is used to quantify and preserve uncertainties on situation recognition. To implement and illustrate the application layer, Java data structures and Operating Systems utilities are used.

To implement the dynamism and distributed features of KoDA, event-driven and network programming techniques are adopted. In event-driven programming, procedures are automatically executed based on predefined actions or events such as the users' and sensors' inputs. Network programming exploits network resources to decentralise computing systems. The event-based programming is adopted to continuously monitor the environment. The network programming is adopted to facilitate communications of software components between computers. A synthesis of the two techniques enables the prototype to (i) dynamically respond to changes within an environment, (ii) to support different environments and (iii) to support resource-constrained devices.

[1] http://www.w3schools.com/xml/xml_whatis.asp
[2] http://jaxp.java.net/
[3] http://research.microsoft.com/en-us/um/redmond/groups/adapt/msbnx/msbn3/msbn3.htm

To evaluate KoDA, experiments were conducted. The data was gathered automatically by the prototype and manually by participants. To facilitate manual data gathering, this research adopts *Experience Sampling Method* and in particular the *Experience Sampling Forms* (Csikszentmihalyi & Larson, 1992; Larson & Csikszentmihalyi, 1983). The resultant datasets are available for download at `http://www.ahisec.com/research`. To assess the probability that the differences between the outcomes of different experiments could be obtained by chance, the *Fisher's test* is used. To assess the probability that the outcome of a set of the experiments could be obtained by chance, the *binomial test* is used.

1.7 Research Contributions and Limitations

This section summarises the contributions of this research and their limitations.

1.7.1 Synthesised Taxonomy of Context Parameters

The synthesised taxonomy of context parameters, appendix A, is an enhancement of the existing taxonomies of context parameters. This taxonomy provides a systematic way of identifying and representing knowledge about common entities in Context-Awareness. This taxonomy is comprehensive as it is developed without an influence of the development of any context-aware system.

1.7.2 Knowledge-intensive Context Model

The Knowledge-intensive Context Model (K*i*CM), figure 4.1, is a novel model of a situation. It is developed to systematically identify and represent knowledge

about entities required to sufficiently represent a situation. K*i*CM improves the existing context models by including knowledge about more entities that are essential for describing an occurrence of a situation. K*i*CM also allows a few of the numerous context parameters of each of the entities to be used when designing a context-aware architecture.

Like any other model, K*i*CM is also an approximation of the real world. Hence K*i*CM does not provide a mirror image of real world situations. K*i*CM only combines relevant knowledge about relevant entities to represent the world in which the users and devices interact. In particular, it combines knowledge about the users and their computer-related activities, location, time and devices to approximate situations. This implies that the knowledge about situations that KoDA possess is intensive but incomplete.

1.7.3 Knowledge-driven Distributed Architecture

The Knowledge-intensive Distributed Architecture (KoDA), figure 5.1, is a novel context-aware architecture designed to dynamically recognise ongoing situation. KoDA enables context-aware systems and subsequently devices to be effectively and intelligently used. KoDA improves the existing context-aware architectures as it is designed based on K*i*CM. Hence, KoDA is developed with a comprehensive model of the real world in which the users and devices interact. Thus, KoDA can use available information to recognise ongoing situation.

In KoDA, however, knowledge about situations is encoded by a developer. Hence the ability of KoDA to recognise ongoing situation is limited to this knowledge. KoDA cannot infer new situations from the knowledge it possess. In addition,

since inference rules are specified based on context parameters, which some of them are interpreted from data gathered by sensors, adding or removing a sensor from KoDA requires a modification of these rules.

1.8 Research Dissemination

This section provides a list of publications related to this research.

1. **Dennis Lupiana**, Rose Tinabo, Fredrick Mtenzi, Ciaran ODriscoll and Brendan OShea, *"Alphanumeric Data: Minimising Privacy Concerns in Smart Environments"*, International Journal of Digital Society (IJDS), Volume 2, Issue 3, 2011.

2. **Dennis Lupiana**, Ciaran O'Driscoll and Fredrick Mtenzi. *"Defining Smart Space in the Context of Ubiquitous Computing"*, Ubiquitous Computing and Communication Journal (UbiCC), Vol 4, Special Issue on Web and Agent Systems, pp 516 - 524, 2009.

3. **Dennis Lupiana**, Ciaran O'Driscoll and Fredrick Mtenzi, *"Characterising Ubiquitous Computing Environments"*, International Journal of Web Applications, Vol 4, Issue 4, pp 253 262, 2009.

4. **Dennis Lupiana**, Fredrick Mtenzi, Ciaran O'Driscoll, and Brendan O'Shea, *"Strictly alphanumeric data: Improving privacy in smart environments"*, In Proceedings of the 5th International Conference for Internet Technology and Secured Transactions, pp 1 3, 2010.

5. **Dennis Lupiana**, Zanifa Omary, Fredrick Mtenzi, and Ciaran O'Driscoll,

"Smart Spaces in Ubiquitous Computing", In Proceedings of the 4th International Conference on Information Technology, 2009.

6. **Dennis Lupiana**, Ciaran O'Driscoll and Fredrick Mtenzi, *"Taxonomy for Ubiquitous Computing Environments"*, In Proceedings of the 1st International Conference on Networked Digital Technologies, pp 469 - 475, 2009.

1.9 Thesis Structure

Chapter 2 provides a background on the Context-Awareness. This chapter provides a survey of the existing context-aware systems and taxonomies of context parameters. This chapter uses the insights gained from the survey to develop a synthesised taxonomy for context parameters. This chapter concludes that the majority of the existing context-aware systems are responsive to individual or a few context parameters rather than to the users' ongoing situation.

To understand the problem, the analysis of the state-of-the-art of their supporting architectures is provided in chapter 3. This chapter concludes that the existing architectures are designed based on inadequate context models. The majority of these models are not developed to model a situation. As a result, the existing architectures are designed with little or no knowledge about situations to effectively use available information to recognise them.

To address the limitations of the existing context models, chapter 4 discusses how this research extends the existing work in context models to develop a novel generic and Knowledge-intensive Context Model (K*i*CM). This chapter discusses how this research leverages a socio-technical theory, the *Actor-Network Theory*,

the Semantic Network and the synthesised taxonomy of context parameters to develop this model.

To address the limitations of the existing context-aware architectures, chapter 5 discusses how this research leverages K*i*CM to design Knowledge-driven Distributed Architecture (KoDA). This chapter outlines the design requirements and discusses each component of KoDA. This chapter also provides a structural representation of the implementation of KoDA.

To illustrate the capabilities of KoDA for supporting context-aware systems, chapter 6 provides a description of the implementation of the prototype. This chapter shows how K*i*CM can be used to represent knowledge about situations. This chapter also shows how KoDA can continuously monitor a real environment and dynamically utilise the acquired information to recognise ongoing situations. This chapter also illustrates how KoDA can be used in the real world.

Chapter 7 provides a description of how the performance of KoDA has been evaluated. In particular, this chapter describes the evaluation criteria, the adopted statistical tests, the design of the experiments, how the data was gathered and the dataset was created, and how the experiments were conducted. This chapter also provides analysis of the gathered data and its interpretations. This chapter concludes by drawing conclusions from the discussion of the results.

A summary and conclusion of the research is provided in chapter 8. The summary and benefits of the contributions from this research are provided. This chapter also outlines limitations and projects the future directions of the research in context-aware architectures and in Context-Awareness research community.

Chapter 2

Context Awareness

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it" (Weiser, 1991). This excerpt implies that the ultimate goal of Ubiquitous Computing (UbiComp) is to make devices *invisible*. This goal has two implications; (i) devices should be abundant, everywhere and readily available and (ii) the use of devices should be intuitive.

This research focuses on the second implication and hence this chapter provides an extensive background on Context-Awareness. The chapter answers some key questions about UbiComp and shows how Context-Awareness relates to UbiComp. This chapter provides a survey of the existing context-aware systems and taxonomies of context parameters. This chapter uses the insights gained from these surveys to develop a synthesised taxonomy of context parameters.

2.1 Introduction

The Ubiquitous Computing (UbiComp) paradigm has inspired the invention of numerous devices. Although these devices offer the users many convenient ways to accomplish their everyday tasks, it remains a challenge for the users to use them effectively. This is more challenging as devices are mobile. The users' working environments become *open* and hence less predictable. The users and their devices enter and leave different working environments where different settings and computing needs may be required. This makes interacting with devices difficult and more time consuming.

In response to these challenges, a *Context-Awareness* research strand emerged. The main focus of this strand is to investigate different principles, methodologies and techniques required to develop system software that can adapt to their dynamic environments and the users' computing needs. These systems are called *context-aware systems*. Initial context-aware systems used location or identity information to automatically provide users' computing needs. To date there are many context-aware systems, each exploiting different aspects of the real world.

Section 2.2 provides a background of the UbiComp paradigm, highlighting its different research strands. This section also explains how research in Context-Awareness fits under the rubric of UbiComp. Section 2.3 provides a background of Context-Awareness followed by an analysis of the existing context-aware systems in section 2.4. Section 2.5 provides an analysis of the existing taxonomies of context parameters and describes a synthesised taxonomy of context parameters. Section 2.6 provides a summary and conclusion remarks.

2.2 Ubiquitous Computing

UbiComp is a computing paradigm that aims to inspire engineers and researchers from different disciplines to invent *human-centred* devices (Weiser, 1991). Weiser and his colleagues envisaged a *world* where people interact and use hundreds of devices without consciously being aware of them. He argued that for devices to be effectively utilised, they should *disappear* from the users' everyday life.

This vision has two implications; (i) devices should be abundant, everywhere and readily available and (ii) devices should be intuitive. To address these challenges, different research strands emerged. This research categorises the strands into Computing *Everywhere* and *Invisible* Computing. The rest of this section provides a discussion on each of these categories.

2.2.1 Computing *Everywhere*

One of the goals of UbiComp is to enable the pervasive availability of computing power. To achieve this goal, research in this strand develops new devices - not as a single-box with mouse and keyboard attached to it. This strand is characterised by devices of different sizes and shapes, and most importantly devices that are capable of *knowing* their locations (Weiser, 1991). These devices are either mobile or stationary. The pioneers in this research strand are Streitz *et al.* (2005) and Russell *et al.* (2005).

UbiComp intends to extract computing power from a *"single box"* and distribute it into independent and yet interconnected devices. This is a new era

of distributed computing. Unlike in traditional distributed systems, where computing power is only available in predefined standalone terminals, computing power in UbiComp is readily available almost everywhere. In this era, the users are supported by multiple devices; some are wearable or handheld while others are embedded or scattered almost *everywhere* in physical environments.

In this strand, the focus is on exploring different design principles, methodologies and techniques to develop devices of different shapes, sizes and functionalities. The strand also seeks to enhance connectivity technologies in order to facilitate cooperation between devices. Although some of these devices are expensive, the majority are reasonably inexpensive. This allows people to own more than one device and hence increases their accessibility. Interconnected by wireless and wired networks, these devices cooperate with each other to effectively support the users and hence facilitate pervasive availability of computing power.

It is still a challenge, however, to precisely quantify computing everywhere. In particular, what distance interval should be considered in order to conclude that computing power is everywhere? Considering previous computing paradigms, however, the availability of computing power in UbiComp should not be limited to a single environment. If computing power is not in an entire city, country, or world then it should be available at least in different environments in a building. This scope, which this research is focusing on, is much more realistic. The infrastructures to support connectivity between the devices are already in place, and the users and their devices operate within similar social settings.

2.2.2 *Invisible* Computing

Another goal of UbiComp is to make devices *invisible*. Its pioneers, Weiser and colleagues, envisaged a computing era where the users interact with and use hundreds of devices with no or less disruptions. Invisibility, however, is a well established concept in Computer Science. In mainframe computers, for instance, the users access computing power remotely without consciously knowing where it come from. In Internet systems also, numerous servers are accessed in response to a single search request. If invisibility of computing power is not a new concept, then what is unique about it in UbiComp?

Invisibility in UbiComp comes with different scope and magnitude. The increasing number of devices and their mobility make computing environments *open* and hence devices operate in dynamic environments. Unlike other computing paradigms where the operating scope of devices is well defined and static, computing environments in UbiComp are dynamic and less predictable. In UbiComp also numerous devices interact in support of the users. In this regard, invisibility in UbiComp focuses on the disappearance of the abundant availability of devices that are in support of the users (Buxton, 1997).

This kind of invisibility, which this research focuses on, is referred to as *psychological* or *mental* disappearance (Russell *et al.*, 2005; Weiser, 1991). In this invisibility, the users use devices subconsciously. In this invisibility, context-aware systems play a significant role. A discussion on context-aware systems is provided in section 2.4. This invisibility, however, cannot be effectively achieved by a standalone context-aware system. Hence, various context-aware systems

should be used to create a computing environment where interactions between the users and devices are seamless and devices are less intrusive.

The *physical* invisibility of devices also plays a significant role in realising psychological disappearance. Russell *et al.* (2005) refer to any invisibility resulted from the physical appearance of devices as a physical invisibility. Devices in UbiComp come in different shapes and sizes in order to make them more accessible (Weiser, 1991). It is through these features that the users can use wall-size displays, wearable and handheld devices subconsciously. Hence, the psychological invisibility is not because these devices cannot be seen or touched, but because they come in different forms than the users know.

Two decades have passed since Weiser and his colleagues envisaged computing environments where devices will be everywhere and unobtrusive to effectively support the users in their daily routines. The question is, *Are we there yet?*. To address this question, a survey of the existing context-aware system is provided in section 2.4. Prior to this survey, a background on context-awareness is required. This background is provided in section 2.3.

2.3 Context-Awareness

The philosophy behind context-awareness was originally introduced in UbiComp by Weiser (1991). He asserted that information within a device's proximity can play an important role into making the use of devices intuitive. He argues that if a device knows its location and surroundings it can adapt its behaviour in significant ways. According to him, location is a physical environment rather

than merely a piece of information such as a name or geographical coordinates. This is further elaborated when Weiser (1994) argued that understanding of people's surroundings is the key factor into making devices *invisible*.

Conforming to Weiser, Schilit *et al.* (1994) envisaged systems that examine and adapt to the user's dynamic contexts. They refer to such systems as *context-aware systems*. They argue that as the users and their devices move, their contexts change. Hence, these systems are meant to seamlessly adjust their behaviours and provide or gather information in response to their environments and the users' computing needs. Schilit *et al.* (1994) argue that *"the interesting part of the world around us is what we can see, hear and touch"*. They argue that information about location, people you are with, nearby devices, environment conditions and time is crucial to context-awareness.

Context-aware systems have a significant impact on how the users perform their daily activities. Imagine walking to a room where a meeting is in progress and your mobile phone automatically switches to a silent mode and back to its normal settings immediately after the meeting is finished. Imagine also walking into a lecture room and a projector in the room automatically switches on and displays your presentation slides. These scenarios, and many more, exemplify how context-aware systems can be useful. The question is, *How would a device know the users' situations in order to appropriately respond?*. This is the question which this research aims to address.

Initial context-aware systems are described to directly respond to context parameters. Weiser (1991), for instance, describes systems that automatically open doors to the right badge wearers, greet people by their names and forward calls

to the users' locations. Schilit and Theimer (1994) describe a system that can automatically provide the users with information about their current location. To date, many context-aware systems have been developed. A survey of these systems is provided in section 2.4.

2.4 Survey of Context-Aware Systems

A context-aware system is a computer system that can appropriately respond to the user's *context* (Brown *et al.*, 1997; Harter *et al.*, 2002; Schilit *et al.*, 1994). There is a number of surveys conducted on context-aware systems but with different focus. Chen and Kotz (2000) focused on identifying context parameters that are commonly used. Baldauf *et al.* (2007) focused on architectural solutions while Hong *et al.* (2009) focused on classifying context-aware research and systems. This survey focuses on identifying how the term context is used in the existing context-aware systems. As part of this survey, a classification of the existing context-aware systems is provided.

Different context-aware systems provide different functionalities and hence are categorised differently. Schilit *et al.* (1994) categorise these systems based on their ability to provide information or execute computing services. They categorise them as *proximate selection*, *automatic contextual reconfiguration*, *contextual information and commands* and *context-triggered action*. Similarly, Dey and Abowd (2000a) categorise these systems by their ability to present information and services, execute services or facilitate acquisition and management of information for later retrieval. They categorise them as *presentation*, *execution*

and *tagging*. Likewise, this research classifies the existing context-aware systems based on their functionalities.

Schilit *et al.* (1994), and Dey and Abowd (2000a) refer to the systems that provide information and list of services as *proximate selection* and *presentation* respectively. Dey and Abowd (2000a) also categorise the systems that facilitate the gathering of information as *tagging*. These systems, however, deliver or gather some kind of content, whether in a text, audio or video format. Hence, this research prefers to use generic terms and refers to these categories as *Content Delivery Systems* and *Content Gathering Systems* respectively. The rest of the categories from Schilit *et al.* (1994) and the *execution* category from Dey and Abowd (2000a) are referred to as *Service Triggering Systems*.

The context-aware systems that are considered in this survey are those that exhibit some kind of autonomy. This survey also takes into account only *contexts* that trigger the systems' responses. If a context-aware reminder, for instance, takes the identity of the creator, the time when the reminder was created and the locations where the reminder should be triggered, then only location is regarded as a *context*. This is because the reminder will only be triggered at the specified location. The information about the creator and time the reminder was created only provide details of the reminder.

Table 2.1 provides a classification of the existing context-aware systems. The first four columns represent the classes of these systems and the rest represent different context parameters used. In this research, *temporal* is preferred over *time* because it is not limited to time only. BioSignals is used to refer to any internal state of the users such as mood.

Table 2.1: Survey of Existing Context-Aware Systems

	SeTS	CoDS	CoGS	*Id*	*Rol*	*Loc*	*Occ*	*Tem*	*Act*	*Gest*	*BioS*
Lee & Cho (2013)	x	x	✓	✓	✓	✓	x	✓	x	x	x
van de Westelaken *et al.* (2011)	x	✓	x	x	x	x	x	x	x	✓	x
Baltrunas *et al.* (2011)	x	✓	x	✓	x	✓	x	✓	x	x	✓
Liu (2010)	x	✓	x	x	x	x	x	x	x	x	✓
Kukkonen *et al.* (2009)	x	x	✓	x	x	✓	x	✓	x	x	x
Adomavicius & Tuzhilin (2008)	x	✓	x	✓	x	✓	x	x	x	x	x
Froehlich *et al.* (2007)	x	x	✓	x	x	x	x	x	✓	x	x
Ludford *et al.* (2006)	x	✓	x	x	x	✓	x	✓	x	x	x
Sohn *et al.* (2005)	x	✓	x	x	x	✓	x	x	x	x	x
Raento *et al.* (2005)	x	x	✓	x	x	✓	x	✓	x	x	x
Intille *et al.* (2003)	x	x	✓	x	x	✓	x	✓	x	x	x
Dey & Abowd (2000b)	x	✓	x	x	x	✓	x	x	x	x	x
Cheverst *et al.* (2000)	x	✓	x	✓	x	✓	x	x	x	x	x
Marmasse & Schmandt (2000)	x	✓	x	x	x	✓	x	✓	x	x	x
Dey *et al.* (1999)	x	✓	x	✓	x	✓	x	x	x	x	x
Abowd *et al.* (1997)	x	✓	x	✓	x	✓	x	✓	x	x	x
Schilit & Theimer (1994)	x	✓	x	✓	x	x	x	x	x	x	x
Want *et al.* (1992)	✓	x	x	✓	x	✓	x	x	x	x	x

Key:

SeTS - Service Triggering Systems
CoDS - Content Delivery Systems
CoGS - Content Gathering Systems
Id - Identity
Rol - Role
Loc - Location
Occ - Occupancy
Tem - Temporal
Act - Activity
Gest - Gesture
BioS - BioSignals

2.4.1 Service Triggering Systems

As discussed by Schilit *et al.* (1994), and Dey and Abowd (2000a), context-aware systems in this category can automatically execute computing services. In such systems, a context is used as an essential input to automatically trigger a certain computing service. In the *Active Badge* location system (Want *et al.*, 1992), for instance, information about location and identity of call recipients is used to automatically forward a call to the recipients.

As noted in table 2.1, little has been done in this category and therefore there are many research opportunities. As part of this research, as will be discussed in chapter 6, a system that remotely and automatically switches ON or OFF the users' computers is illustrated. The system switches a computer ON or OFF depending on a recognised situation. The Mouse Activity Monitor and Keyboard Activity Monitor, which are also developed as part of this research, also belong to this category. These systems remotely and automatically execute services to determine mouse and keyboard status of the users' computers respectively.

From a Computer Science perspective, delivering or gathering of content is subject to the execution of certain computing services. As is evident from the surveyed context-aware systems, however, in this category the definition of computing service is limited to only the services which affect the behaviours of devices. Therefore, context-aware systems that can assist the users by gathering or delivering some kind of content are discussed separately.

2.4.2 Content Gathering Systems

In this category, context-aware systems support the users by automatically gathering information. Likewise, a context is used as an input and hence these systems directly respond to context parameters. As shown in table 2.1, the common context parameters in this category are location (Kukkonen *et al.*, 2009), time (Kukkonen *et al.*, 2009) and activity (Froehlich *et al.*, 2007). The existing context-aware systems in this category include classroom multimedia notes generating (Abowd, 1999) and experience capturing systems (Froehlich *et al.*, 2007; Intille *et al.*, 2003; Kukkonen *et al.*, 2009; Raento *et al.*, 2005).

2.4.3 Content Delivery Systems

Context-aware systems in this category automatically provide a content to the users. Likewise, a context is used as an input and hence these systems directly respond to context parameters. As shown in table 2.1, the common contexts in this category are identity (Cheverst *et al.*, 2000), location (Adomavicius & Tuzhilin, 2008), time (Baltrunas *et al.*, 2011), gesture (van de Westelaken *et al.*, 2011) and biosignal (Baltrunas *et al.*, 2011; Liu, 2010). The existing context-aware systems in this category include reminder systems (Ludford *et al.*, 2006; Sohn *et al.*, 2005), tour guide systems (Abowd *et al.*, 1997; Cheverst *et al.*, 2000) and recommendation systems (Liu, 2010; van de Westelaken *et al.*, 2011).

As noted in table 2.1, the content to be delivered can be of any type. For instance, a system that provides map information to the tourists or a system that reminds the users are both providing the users with content. The tour

guide system provides text, image and probably audio whereas the reminder system may provide text or audio. As shown in table 2.1, the content delivered by the systems in this category include text (Abowd *et al.*, 1997; Cheverst *et al.*, 2000) and audio (Liu, 2010; van de Westelaken *et al.*, 2011). As in the previous category, a context is used as an essential *input* to automatically locate and provide the users with appropriate content.

Brown *et al.* (1997) categorise the systems in this category as *continuous* and *discrete* systems. They define a continuous system as a system that constantly provides content to the users while a discrete system as a system that provides content to the users only when a certain context parameter is met. They exemplify a tour guide system, where location information is constantly provided to the tourists' mobile devices, as a continuous system. Although it is plausible to categorise such a system as a continuous system, its operations are limited to a certain location. Hence, like discrete systems, the content delivered to the users by continuous systems depends on a context parameter.

As is evident from the survey, the majority of the exiting context-aware systems directly respond to a specific context parameter. Only few of these systems respond to a number of context parameters. Baltrunas *et al.* (2011), for instance, aggregate various context parameters to recommend musics to listeners. In these systems, a set of context parameters is used as a condition required for a task to be accomplished. In these systems, a task is regarded as a situation (Dey, 2000; Henricksen, 2003). While these systems are more responsive to their environments, they do not exploit the richness of information about their environments to adapt to social dynamics.

With the increasing number of devices and their mobility, computing environments become *open* and hence less predictable. The users enter and leave different environments while using devices. The users also constantly interact with their colleagues, who may have different social relations, in their daily routines. Devices also interact with other heterogeneous devices. Furthermore, time and physical properties of different environments do change. As a result, the users and hence their devices operate in dynamic environments. Hence, it is important for context-aware systems to be responsive to users' ongoing situations rather than to one or more context parameters that are specific to a task.

2.5 Taxonomies of Context Parameters

To enumerate context parameters, various taxonomies of context parameters are developed. This section provides a survey of the existing taxonomies of context parameters and proposes a synthesised taxonomy of context parameters. This taxonomy provides a list of common entities and their context parameters. As discussed in chapter 4, this list provides a basis for developing a knowledge-intensive model of a situation. A complete list of entities required to develop this model is provided in section 4.5.1.

The aim of this survey is to identify an extensive list of common context parameters without being influenced by a specific domain or the development of a context-aware system. This survey also seeks to identify key entities in this research domain. To consolidate the identified parameters, this research develops a synthesised taxonomy. Section 2.5.1 provides a survey of the existing

taxonomies of context parameters.

2.5.1 Survey of the Existing Taxonomies

Schilit *et al.* (1994) identify information about "where you are", "who you are with" and "what resources are nearby" as important for developing context-aware systems. Schilit (1995, p. 12 - 19) argues that such information is dynamic and hence categorises it as *communication*, *environmental* and *location* dynamics. According to him, communication dynamics are changes associated with communications, such as bandwidth and status of devices. Environmental dynamics are changes, such as physical conditions, people and their *social situations*. From the user's point of view, he defines location dynamics as changes of physical locations as the user enters different environments.

Schmidt *et al.* (1999b) categorise context parameters into *human* and *physical environmental*. Human parameters are sub-categorised into *user*, *social environmental* and *task*. Similarly, the physical environmental parameters are sub-categorised into *physical condition*, *infrastructure* and *location*. User parameters refer to information about a particular user such as habits and emotion. Social environmental parameters refer to information about nearby users and their social relations. Task parameters refer to goals and activities the user is involved in. Physical parameters refer to physical properties of an environment such as noise, temperature and light. Infrastructure parameters refer to information about computing resources such as computers, displaying units and network devices. Location parameters refer to information about an environment.

Dix *et al.* (2000) categorise context parameters into *infrastructure*, *system*, *do-*

main and *physical*. According to Dix *et al.* (2000), infrastructure parameters refer to information about computing resources which are essential for facilitating computing mobility. These computing resources include, but are not limited to, network devices. System parameters refer to computing entities which are part of a context-aware system and their functionalities such as feedback mechanisms. Domain parameters refer to semantics of the system domain such as consistent definitions of common terminologies and relationships between the users and their mobile devices. Physical parameters refer to physical spaces of a mobile host and a physical environment where the mobile host resides.

Mitchell (2002) categorises context parameters into *identity*, *spatial*, *temporal*, *environmental*, *social*, *resource*, *resource management*, *goal/tasks* and *user history*. Identity refers to information about identity of a user whereas spatial parameters refer to information related to location, orientation and velocity. Temporal parameters refer to information related to time while environmental parameters refer to physical properties of an environment such as light, noise and temperature. Social parameters refer to information about nearby users whereas resource parameters refer to information about nearby resources. Resource management parameters refers to information about resources availability and usage. Goal/task parameters refer to information about scheduled or occurring tasks while history parameters refer to previous user activities and interactions.

Göker and Myrhaug (2002), focusing on the users, categorise context parameters into *environmental context, personal context, social context, task context* and *spacio-temporal context*. Environmental parameters refer to information that can be observed about a physical environment such as existing users, physical

properties and computing infrastructure. The personal context parameters are further categorised into *physiological* and *mental*. Physiological parameters refer to attributes of a user such as blood pressure, hair colour and weight whereas mental refers to attributes of user such as emotion, stress and mood. Social parameters refer to social relations between users such as colleague, supervisor or leader. Task parameters refer to information about users and nearby users activities such as goals, actions or events. Moreover, spacio-temporal parameters refer to information related to time and location.

Han *et al.* (2008) categorise context parameters into *physical, internal* and *social*. Han *et al.* (2008) refer to physical parameters as information about nearby users and any relevant physical objects. Referring to the users, they exemplify internal parameters as emotion, goals and thoughts. Similarly, Han *et al.* (2008) refer to social parameters as information about social relations and social phenomenon. Han *et al.* (2008), however, relate each category of context parameters to timeliness. They argue that each context belongs to certain time frames and therefore time also plays an important role in realising context-awareness.

Soylu *et al.* (2009) categorise context parameters into *user, device, system, information, environmental, time, historical* and *relational*. User parameters are sub-categorised into *external user* and *internal user*. Similarly, device parameters are sub-categorised into *hard device* and *soft device*. In addition, environmental parameters are sub-categorised into *digital environmental* and *physical environmental*. External parameters refer to physical properties of a user such as identity while internal parameters refer to properties such as emotion. Hard device parameters refer to physical properties of devices such as display while

soft device parameters refer to software components. system parameters refer to capabilities and system requirements of an system. Information parameters refer to properties of information to be shared. Digital environmental parameters refer to computing entities such as network devices while physical environmental parameters refer to physical entities. Time parameters refer to different attributes of time such as hour, day and season. Historical parameters refer to previous occurred situations while relational parameters refer to information about relations between different context categories.

2.5.2 A Synthesised Taxonomy of Context Parameters

It is evident from the survey, section 2.5.1, that the users, locations, time and computing resources are the most common entities. Therefore, this research develops the taxonomy based on these entities. In this research, however, the specific terminologies are replaced by the generic terminologies. Therefore the user, location and time are replaced by personal, spatial and temporal respectively. Figure 2.1 shows the taxonomy with context parameters that are used for implementing a prototype. Appendix A provides a complete synthesised taxonomy of context parameters. The rest of this section discusses each of the categories of context parameters.

- **Personal Context Parameters**

In this taxonomy, any information that can characterise a person is categorised as a *personal context parameter*. Since people can be characterised by their physical and psychological attributes, the category is sub-categorised into *physical parameters* and *psychological parameters*. Since people belong to a physical

world and are associated with one or more devices, their relationships should be established. Likewise, since people are part of a certain social community, the relationships among them should also be established. In order to accommodate these relationships, the taxonomy includes a *relational parameters* subcategory.

Similar categorisation can be observed from the existing taxonomies. Schmidt *et al.* (1999b), for instance, categorise user parameters as part of human context parameters. Schmidt *et al.* (1999b) describe user parameters as information about habits and emotion of a person. Han *et al.* (2008) and Soylu *et al.* (2009) also categorise emotion, thoughts and goals as internal parameters. Likewise, Göker and Myrhaug (2002) categorise personal context parameters to physiological parameters and mental parameters. Since both parameters describe internal attributes of a person, they fit under the psychological parameters sub-category. Mitchell (2002) identifies user's identity as a context parameter which fits under the physical parameters sub-category.

- **Temporal Context Parameters**

As is evident in the survey, few taxonomies include time as one of their categories. Mitchell (2002), for instance, considers temporal information as one of the essential categories of context parameters. Likewise, Göker and Myrhaug (2002) consider temporal information by combining spatial and temporal context parameters in one category. Furthermore, Soylu *et al.* (2009) identify time as one of the crucial categories of context parameters. Therefore, in line with the few taxonomies, this taxonomy categorises any information related to time as *temporal context parameters*.

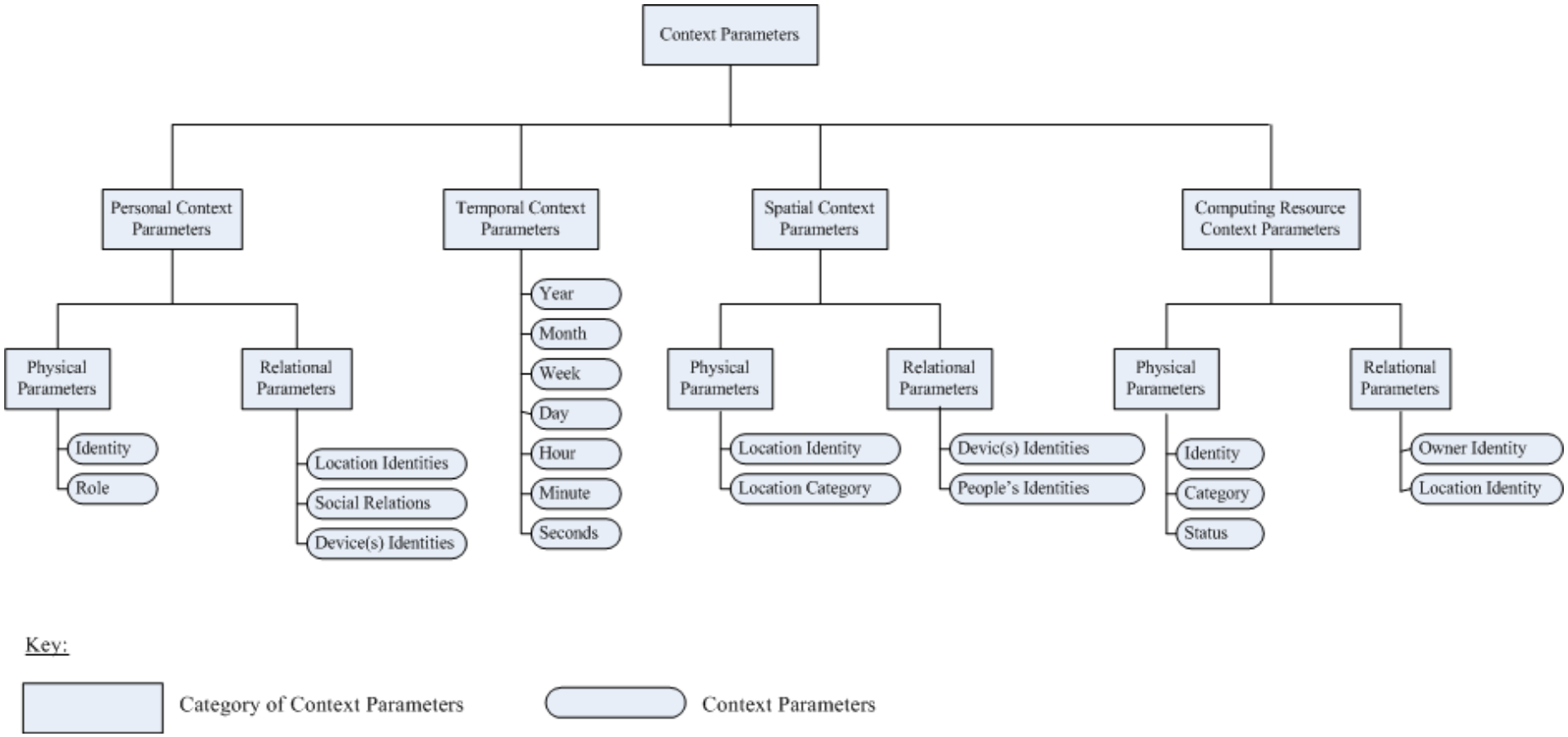


Figure 2.1: A Partial Synthesised Taxonomy of Context Parameters

- **Spatial Context Parameters**

In this taxonomy, any information that can characterise a physical environment is categorised as *spatial context parameter*. Since environments can be characterised by their physical attributes, the category is sub-categorised into *physical parameters*. The users and computing resources are not part of a physical environment. Instead, the users and computing resources are only related to a certain environment. Therefore, a sub-category of *relational parameters* is included in order to accommodate relationships. Unlike physical context parameters, relational context parameters provide information about relationships between physical environments and other relevant entities such as the users and computing resources.

As is evident from the survey, information related to computing resources is widely categorised as *infrastructural parameters*, which is part of *environmental context parameters*. Schilit *et al.* (1994), for instance, categorise information related to computing resources as infrastructural parameters. Similarly Soylu *et al.* (2009) categorise the information in digital environmental parameters, which is similar to infrastructural parameters. Furthermore, Dix *et al.* (2000) categorise information related to computing resources as infrastructure context. Schmidt *et al.* (1999b) also categorise information related to computing infrastructure in the infrastructure parameters. Additionally, Schmidt *et al.* (1999b) sub-categorise information related to the users as part of environment context parameters. As noted in the preceding paragraph, the users and computing resources are not part of a physical environment.

- **Computing Resource Context Parameters**

In this taxonomy, any information that can be used to characterise a computing resource is categorised as *computing resource context parameter*. This research defines a computing resource as any computing entity which can be hardware or software. A computer and a computing service, for instance, are computing resources. Since computing resources can be characterised by their physical attributes, the category is sub-categorised into *physical parameters*. Likewise, since a computing resource belongs to either a person or a particular physical environment, a *relational parameters* sub-category is included.

It is evident from the survey that the majority of the existing taxonomies categorise information related to devices as a sub-category of environmental or spatial context parameters as referred to in this taxonomy. Schmidt *et al.* (1999b), for instance, referring to devices as infrastructure categorise them as a sub-category of a physical environment. Like any other entities, devices can be uniquely identified and each has its unique attributes. Information about relations to a person or a physical environment is represented in the *relational parameters* sub-category.

While these context parameters are essential for developing context-aware systems in different problem domains, they play different roles. To differentiate them, Dey and Abowd (2000a) introduce the concept of *primary context parameters*. This concept is also emphasised by Chen and Kotz Chen & Kotz (2000). Dey and Abowd (2000a) refer to a primary context parameter as a primary key for determining other relevant context parameters of a particular entity. An identity, for instance, can be used to determine the role and gender of a user.

In this taxonomy, the identity context parameters are referred to as *primary*

context parameters whereas the rest of the context parameters are referred to as *secondary context parameters*. As for the temporal context parameters, there is no single parameter to uniquely identify a timestamp and hence each element of time is crucial. This is also true in practice since time can only be differentiated by including all of its elements.

2.6 Summary and Conclusion

This chapter provided the background of Ubiquitous Computing paradigm and discussed research in Context-Awareness where a survey of the existing context-aware systems and the survey of the existing context taxonomies are provided. The survey of the existing context-aware systems found that the majority of the systems are not responsive to ongoing users' situations. The majority of these systems are responsive to individual context parameters. The few existing context-aware systems that respond to more than one context parameters simultaneously are limited to specific tasks.

In literature, it is agreed that context-aware systems are to be used in the natural settings of the users where different entities interact and each entity has an impact on other entities. Such settings occur in environments which are highly dynamic. In addition, the users' computing needs are changing. To effectively support the users in their daily routines, this research argue that social aspects should be taken into account when designing supporting architectures of context-aware systems. Hence, these architectures should be designed with a comprehensive model of the real *world* in which the users and devices interact.

Since entities and their relationships are very important in modelling a situation, identifying an extensive list of their context parameters is also important. This research accomplished this by developing a synthesised taxonomy of context parameters. The taxonomy is comprehensive and generic since it is developed without an influence of a specific system domain or the development of a context-aware system. The taxonomy plays an important role when developing a model of situations. It provides a comprehensive and systematic way of identifying and representing context parameters. Chapter 3 provides analyses of the existing context-aware architectures.

Chapter 3

The State-Of-the-Art

This chapter provides analyses of the existing context-aware architectures and their context models and inference mechanisms. Since architectural solutions emerged to address limitations of the existing context-aware systems, there is no one-to-one matching between the context-aware systems surveyed in section 2.4 and architectures reviewed in this chapter.

The chapter reveals that the existing context-aware systems are not responsive to user's situations. This limitation is because the existing context-aware architectures are designed based on context models which do not model situations. As a result, the architectures fail to intelligently exploit available information to dynamically recognise ongoing situations.

3.1 Introduction

Context-aware systems are essential for increasing seamless interactions between the users and devices and for making these devices less intrusive. As the number of devices increases and as the majority are mobile, users and their devices operate in dynamic environments. Hence, context-aware systems should be adaptive to these changes. Context-aware systems should be aware when a situation of a user changes to appropriately respond. However, the majority of the existing context-aware systems are not responsive to users' ongoing situations. Therefore, the question that this chapter addresses is why the majority of the existing context-aware systems are not responsive to users' ongoing situations?

To address this question, a review of the existing architectures is important because they enable context-aware systems to understand their environments and the users' computing needs. Central to these architectures are context models and hence their review is also important. These models provide a basis for representing knowledge about the real *world* in which the users and devices interact. The architectures, through this knowledge and their reasoning capabilities, enable the systems to understand their environments and the users' computing needs. This chapter also provides analyses of the existing inference mechanisms.

Section 3.2 provides an analysis of the existing context-aware architectures while identifying their strengths and limitations. Section 3.3 provides an analysis of the existing context models while identifying their strengths and limitations. The analysis of the models, however, is limited to the models whose architectures have been discussed in section 3.2. The analysis of the existing inference

mechanisms is provided in section 3.4. Section 3.5 provides a discussion of the analyses while highlighting their limitations. Section 3.6 provides a summary and conclusion remarks of the chapter.

3.2 Analysis of Architectures

Research in Context-Awareness has proposed various context-aware architectures to support context-aware systems. This section provides an analysis of the existing architectures.

3.2.1 Initial Architectures

The initial research (Cooperstock *et al.*, 1997; Harter *et al.*, 2002; Shafer *et al.*, 1998) has utilised context-aware systems to automate the user's repetitive computing tasks in a particular environment such as an office, a meeting room or a classroom. The designs of their supporting architectures emphasise monitoring the environments and interpreting data acquired by sensors. Cooperstock *et al.* (1997), for instance, utilise motion detectors to monitor the users' movement and utilises the acquired data to automate conference appliances in a conference room. Little effort is taken to effectively utilise available information about relevant entities in the environments to recognise ongoing situations.

The principal aim of these architectures is to reduce physical interactions between the users and devices in predefined settings. As a result, these architectures are designed with *binary relationships* between inputs, which are data from sensors, and the typical computing services that may be required. As will

be discussed in section 3.3.1, a context model is regarded as a representation of knowledge about an entity and in particular information about its specific attribute. Thus, context models are used for data interpretation rather than for recognising ongoing situations. These architectures are designed with the assumption that the users' working environments are *static* and thus their computing needs remain *uniform*, regardless of the changes in the environment.

3.2.2 Context Broker Architecture

Chen (2004) identifies semantic limitations, among others, as the major shortfalls of the initial architectures. He argues that these architectures utilise context models which are semantically poor and poorly represented. He contends that these models are limited to a specific attribute of a particular entity. As a result, the knowledge about the entities is limited and is represented as *objects* of a specific implementation language. These limitations inherently make the initial architectures incapable of knowledge reasoning and sharing. To address these limitations, he designs a *Context-Broker Architecture (CoBrA)*.

Unlike the initial architectures, Chen (2004) focuses on representation, reasoning and sharing of context parameters. He argues that the representation of context parameters is important in order to facilitate interpretation of data acquired from sensors. He also contends that reasoning of such information is important in order to interpret, detect and resolve inconsistencies. Since it is rare for a context-aware system to have a complete knowledge about its surroundings, he also asserts that sharing of context parameters is crucial.

CoBrA, however, is not designed to utilise context parameters to recognise situ-

ations. Like many other context management architectures, such as de Andrade (2007) and Gomes *et al.* (2010), CoBrA is designed to gather and share context parameters with context-aware systems. Hence, a context model in CoBrA is used for interpreting data from sensors, detecting and resolving inconsistencies. Although CoBrA can infer other knowledge, its inference capabilities are limited. It is unable to establish relationships between different entities to recognise ongoing situations. Thus CoBrA is also designed with the assumption that the users' surroundings are static and thus their computing needs are uniform.

3.2.3 Case-based Multi-agent Architecture

Inspired by the role of *context* in human reasoning, Kofod-Petersen (2007) argues that for a context-aware system to automatically provide user-tailored computing needs, it should be able to determine the user's ongoing situation. He defines a situation as a social setting, such as a meeting, where the users want to achieve various goals. He argues that such capability can be realised by an architecture that can *perceive* and *reason* about relevant available information from an environment. To realise these capabilities, he designs an architecture, henceforth referred to as a *Case-based Multi-agent Architecture (CaMA)*.

Unlike the initial architectures and CoBrA, CaMA is designed to take into account information about different entities within an environment to determine ongoing situations and hence to identify the user's goals. Nonetheless, CaMA is designed based on a limited model of the real world and hence recognises situations based on limited knowledge about people, location and time. It is designed based on a limited context model, as will be discussed in section 3.3.3.

Its *context structures*, which define the scope that CaMA can perceive its surroundings, is built based on this model. As a result, CaMA relies on incomplete information and little knowledge about ongoing situations.

3.2.4 Context Engine Architecture

In order to address the limitations of the existing architectures, Kaenampornpan (2009) designs a context-aware architecture, henceforth referred to as *Context Engine Architecture (CEA)*. Like Chen (2004) and Kofod-Petersen (2007), she identifies the semantic limitations as the major shortfall of the existing architectures. She argues that the architectures are designed based on limited context models due to a limited understanding of *context*. Kaenampornpan (2009, p. 55 - 56) asserts that the architectures, as a result, are *application-specific* and hence they cannot be reused and extended.

Kaenampornpan (2009, p. 106 - 107) argues that the main goal of a context-aware system is to support the users and hence the ability to infer their intentions is of paramount importance. To achieve this, she argues that the systems should be able to use implicitly acquired inputs and context. She defines context as a set of interrelated context parameters about the user's activities. She argues that the relationships between the inputs (or context parameters) are also important. To specify the parameters and their relationships, she designs a context model. The discussion of the model is provided in section 3.3.3.

Like CaMA, CEA is designed to take into account information about entities in an environment to infer the user's intentions. CEA is also designed based on a limited model of the real world. It is designed based on a context model which,

as will be discussed in section 3.3.3, is limited. The model specifies relationships between people, locations and time. As a result, CEA relies on limited knowledge limited knowledge about people, location and time. In addition, the model limits the ability of CEA to perceive its surroundings and to sufficiently represent the user's typical situations. As a result, like CaMA, the CEA relies on incomplete information and little knowledge about ongoing situations.

3.3 Analysis of Context Models

To support the design of the existing architectures, various context models have been developed. This section explores the existing models. The aim is to identify any potential research opportunities. This research categorises the existing models into *attribute-based*, *ontology-based* and *theory-based*. The rest of this section discusses each of these models.

3.3.1 Attribute-based Context Models

In order to design context-aware architectures, the initial research exploits discrete information within a user's proximity. Schilit (1995), for instance, uses location to automatically provide the users with location-based computing services. Therefore, according to the initial architectures, a context model is a representation of an entity and its attributes. As a result, much emphasis is on representation of the models into machine-interpretable languages and implementation of appropriate data structures in order to facilitate interpretation of sensed data. Subsequently, the models are implemented as *objects* of implemen-

tation languages (Chen, 2004, p. 4).

Focusing on the processes required to interpret data from sensors, Strang and Linnhoff-Popien (2004) refer to these models as *key-value*. Their analysis, however, is limited on how data from sensors is organised, stored and interpreted. The analysis does not explore how different entities, and their context parameters and relationships within the user's proximity can be used to model the real world. Since much of the structure of these models is based on the notation of attribute and value, this research, like Henricksen (2003), refers to these models as *attribute-based* context models.

The attribute-based context models have two limitations which have a significant impact on designing context-aware architectures; (i) they focus on a specific attribute of an entity and therefore do not take into account the impact of other nearby entities and (ii) they are implemented as *objects* of implementation languages and therefore are integral part of architectures. The former makes the initial architectures semantically poor and therefore incapable of supporting knowledge reasoning and sharing. The latter makes the models language-specific and tightly coupled to implementations and hence incapable of supporting knowledge re-usability and subsequently limits knowledge reasoning and sharing.

3.3.2 Ontology-based Context Models

To abstract the representation of context parameters from any implementation language and to support knowledge reasoning, sharing and re-usability, majority of the researchers adopted Ontology-based approaches. Wang *et al.* (2004), Chen (2004) and Gu *et al.* (2004a), for instance, adopt Ontology-based approaches.

A comprehensive list of the existing ontology-based context models and their detailed discussion is provided by Ye *et al.* (2007). A number of surveys also indicate that Ontology is a promising approach for context modelling (Baldauf *et al.*, 2007; Strang & Linnhoff-Popien, 2004).

Like the attribute-value context models, ontology-based context models organise context parameters based on individual entities but provide more details about these entities. Using Ontology, for instance, relationships between entities of similar types can be specified by using *is-a* relationships. Ontology also specifies different meaning of terminologies used to describe entities and constraints for using these terminologies. Ontologies are represented using Semantic Web knowledge representation languages, such as DAM+OIL and Web Ontology Language (OWL), which are neutral from any implementation languages.

Hence, these models support knowledge sharing and subsequently interoperability. Also through their *is-a* relationships, these models facilitate some kind of knowledge reasoning. These relationships, for instance, enable knowledge about the role of a user to be derived from the information about the user's identity. Nonetheless, ontology-based context models are still insufficient for developing a model of a situation. These models only take into account relationships between entities of similar types. Hence, these models fail to capture relationships between different entities, which are fundamental for modelling situations.

While it is plausible to argue that these models facilitate knowledge reasoning, the answer to this is subjective. The degree of reasoning depends on the extent of intelligence the underlying architecture is sought to demonstrate. If devices are to disappear, as envisaged by Weiser (1991), this research argues that both

knowledge about entities and how they interrelate is required. As argued by Kaenampornpan (2009, p. 75), a true solution to context-aware systems should utilise various interrelated entities. Thus, apart from providing more and consistent details about entities, a knowledge-intensive context model is required.

3.3.3 Theory-based Context Models

Recently, researchers have started to adopt socio-technical theories in effort to provide an abstract representation of the real world. Kofod-Petersen (2007) and Kaenampornpan (2009), for instance, adopt Cultural Historical Activity Theory (CHAT) to model a situation. CHAT is an extension of Activity Theory that provide a theoretical framework for analysing different aspects of human activities in social settings while emphasising on a community (Igira & Gregory, 2009; Kaptelinin & Nardi, 1997). CHAT explores relationships between subject, object, artifact, division of labour, rules and community.

Kofod-Petersen (2007) extends a context model from the AmbieSense[1] project by adding social related context parameters as specified by the theory. His context model includes domain specific values, a copy of CHAT and a model that specifies knowledge required by case-based systems. Kaenampornpan (2009) extends CHAT by including time as part of her model. Since the theory implicitly includes a physical environment, she also adopts location as part of her model.

These models emphasises relationships between different entities of everyday social settings. Nonetheless, the existing theory-based context models are developed to represent different social and physical aspects required in order to

[1] http://ambiesense.co.uk/the_origin/index-1.html

identify and accomplish a user's objective. As a result, these models are limited to relationships between location, people and time. Subsequently, these models limit context parameters to identities of a location and people, and to the people's roles. Kofod-Petersen (2007) also incorporates implementation details on his model. As noted by Newell (1982), a knowledge model should not incorporate any implementation details.

In addition, both of the existing theory-based context models are based on Activity Theory that treats entities differently and hence the relationships among the entities are biased. The theory treats a *subject* as a "super entity". As a result, knowledge about whether nearby user(s) are present or not is used as an input to provide user-tailored computing needs. Hence, knowledge of computer-related activities of nearby users have no impact on situation recognition. This is, however, the opposite in real world environments. The status of devices, the users' computer-related activities, and social relationships between the users have a significant impact on ongoing situations.

3.4 Analysis of Inference Mechanisms

To reason about context parameters, a number of inference mechanisms have been used. This section provides an analysis of the exiting inference mechanisms. This research categorises the existing inference mechanisms into (i) logic-based, (ii) probabilistic and (iii) hybrid inference mechanisms. Since the application of inference mechanisms is broad, this research only considers the existing solutions to context-aware systems. The rest of this section discusses inference

mechanisms in each of these categories.

3.4.1 Logic-based Inference Mechanisms

Logic-based inference mechanisms use prepositional logic to evaluate inference rules. Kofod-Petersen (2007), uses *cases* to represent inference rules but many researchers (Chen, 2004; Dockhorn Costa *et al.*, 2007; Gu *et al.*, 2004b; Lee *et al.*, 2006) use rule-based knowledge representation language. Hence, many researchers have been using the existing rule-based reasoning engines to evaluate inference rules. Chen (2004), for instance, uses Jess[1] rule engine while Dockhorn Costa *et al.* (2007) use a variety of Jess rule engine called DJess. To address uncertainties of context parameters, many researchers for instance Ciaramella *et al.* (2010), Anagnostopoulos and Hadjiefthymiades (2010), and Haghighi *et al.* (2008), use fuzzy logic to implement inference engines.

A number of researchers have been using custom logic-based approaches. Henricksen (2003), and Yau and Karim (2004), for instance, use tables to evaluate inference rules. Since the output of a query is *false* or *true*, Henricksen (2003) introduces another output called *possibly true* in order to deal with uncertainties. This output is achieved by creating additional queries which replace values of one or more conditions of the existing queries with null values. Thus, when the gathered context parameters do not match with any set of conditions of the original queries, alternative queries are used. If one or more of the conditions of the alternative queries matches with the gathered context parameters, then the corresponding situation is said to be possibly true.

[1] http://herzberg.ca.sandia.gov/

3.4.2 Probabilistic Inference Mechanisms

As noted by Henricksen (2003), context parameters are not always complete and their sources are unreliable. Thus, to reason with uncertainties, many researchers have been using the existing techniques from machine learning to develop probabilistic inference mechanisms. Many researchers, Lee and Cho (2013), Santos *et al.* (2011) and Truong *et al.* (2005) for instance, adopt a Bayesian Network (BN). A BN is a directed acyclic graph representing random variables and their causal relationships (Heckerman, 1998; Jensen, 1996). Some researchers use the existing BN inference mechanisms such as MSBNx[1] and EBayes[2]. Biegel (2005) and Ranganathan *et al.* (2004), for instance, use MSBNx and EBayes respectively, as their inference mechanisms.

Other researchers, Li *et al.* (2013), Zhang *et al.* (2010) and Lyu *et al.* (2010) for instance, use Dempster-Shafer theory to develop their inference mechanisms. McKeever (2011) extends Dempster-Shafer theory to include temporal knowledge and quality information of sensors. She extends the operations of the theory to create an evidence decision network. This network specifies processes required to propagate evidence from sensors through a hierarchy of context levels (context values, low level situations and higher-level situations). She defines context values as interpretations of data from sensors. She exemplifies 'leave house' and 'front door used' as higher-level and low level situations respectively. To abstract the processes required to propagate evidence, she uses Directed Acyclic Graph (DAG). She also uses DAG to capture knowledge about situations and to assess

[1]http://research.microsoft.com/en-us/um/redmond/groups/adapt/msbnx/
[2]http://sites.poli.usp.br/pmr/ltd/Software/EBayes/index.html

their belief based on data collected from sensors.

3.4.3 Hybrid Inference Mechanisms

A number of researchers have combined different approaches to develop their inference mechanisms. Gu *et al.* (2004a,b), for instance, illustrate the use of rule-based and Bayesian network inference mechanism in context-aware systems supported by their middleware. Likewise, Ranganathan *et al.* (2004) illustrate the use of rule-based, fuzzy logic and Bayesian inference mechanism. Devlic *et al.* (2009) have illustrated the use of rule-based, Bayesian network and user feedback to infer social relationships between users of the MUSIC middleware (Paspallis *et al.*, 2008). Cimino *et al.* (2012) have combined rule-based reasoning approach, ontology reasoning approaches, fuzzy logic and user feedback to evaluate task-based inference rules. Strobbe *et al.* (2012) have combined rule-based and case-based reasoning approaches to evaluate task-based inference rules.

3.5 Discussion of the Analyses

A context model is a centrepiece of any context-aware architecture. The attribute-based and ontology-based context models, however, are not developed to model a situation. These models are developed to facilitate interpretation of data acquired from sensors. Hence, the emphasis of these models is on representing domain concepts in order to enable context-aware architectures to interpret data from sensors and share the resultant context parameters with context-aware systems. As a result, context-aware architectures lack reasoning capability or

is limited to detecting and resolving inconsistencies. This limits initial context-aware systems to be responsive to individual context parameters.

To remedy this limitation, Dey (2000) and Henricksen (2003) proposed a situation abstraction model in their frameworks. This has become the de facto model in context-aware systems (Lee & Cho, 2013; Liu, 2010; van de Westelaken *et al.*, 2011). In this model, context parameters required for a task to be accomplished are used to specify task-specific inference rules in context-aware systems. This enables context-aware systems to be responsive to more than one context parameters simultaneously. Nevertheless, these parameters are limited to a specific task and hence these systems are unable to respond to social dynamics.

With the situation abstraction model, the role of a context-aware architecture is to acquire and share context parameters with context-aware systems. Inference rules are specified in context-aware systems and hence duplication of the rules is inevitable. Developers are also required to familiarise with knowledge representation languages as these rules are specified by a knowledge representation language. The burden of validating inference rules, which is essentially a knowledge reasoning task, is left to context-aware systems. This approach has serious performance implications to resource-constrained devices and hence a knowledge-driven distributed architecture is required.

In an effort to develop a context-aware architecture that can reason and hence recognise ongoing situations, the theory-based models have been proposed. These models are closer to model real world situations as they emphasise social related aspects and relationships between more than one entity. The existing theory-based models, however, are developed to enable architectures to support context-

aware systems that provide personalised computing needs. These models treat one user as a "super entity" and hence knowledge about whether nearby user(s) are present or not is used to determine the user's goals. Hence, knowledge of computer-related activities of nearby users have no impact.

Gregory (1993) argues that if a model is to play a significant role in developing a system, it should correspond to the real world. He argues that (C_1) the elements of the model should refer to physical or abstract objects of the real world and (C_2) their relationships should have the same logical form as those in the real world. Newell (1982) also argues that (C_3) a knowledge model should not include any implementation details. The attribute-based models fail to comply with C_1 and subsequently with C_2 and C_3. The ontology-based models comply with C_1 but not with C_2 and C_3. Similarly, the existing theory-based models partially comply with C_1 and C_2, and Kofod-Petersen (2007) fails to comply with C_3.

3.6 Summary and Conclusion

The survey of the existing context-aware systems, section 2.4, shows that the majority of these systems are responsive to individual context parameters. This survey shows that only few of these systems are responsive to more than one context parameters that are specific to a task. This chapter provided the analysis of the existing context-aware architectures, their context models and the inference mechanisms with the intention of addressing the question raised in section 3.1. This question sought to find the reasons as to why the majority of the existing context-aware systems are not responsive to users' ongoing situations.

This chapter concludes that the majority of the existing context-aware systems are not responsive to the users' ongoing situations because their supporting architectures are not developed to recognise ongoing situations. Majority of these architectures are developed to facilitate acquisition of data from sensors, interpreting it and share the resultant context parameters with context-aware systems. The few architectures that are developed to recognise ongoing situations are limited to providing personalised computing needs. Hence, these architectures are developed with limited model of the world in which the users and devices interact. Subsequently, these architecture lack reasoning capability or is limited to detecting and resolving inconsistencies.

To remedy the limitations of the existing context-aware architectures and to eliminate or reduce some of the challenges introduced by the alternative solution, knowledge-driven and distributed context-aware architecture is required. This architecture should be designed based on knowledge-rich context model in order for this architecture to be developed with a comprehensive model of the real world. The discussion on how this research designs a knowledge-driven and distributed context-aware architecture is provided in chapter 5. Chapter 4 provides a discussion on how this research extends the work in theory-based context models to develop a knowledge-intensive context model.

Chapter 4

Knowledge-intensive Context Model

This chapter discusses how this research extends the work in the *theory-based* context models to develop a model of situations. The chapter discusses how the Actor-Network Theory (ANT) is adopted to systematically identify and represent the key entities, and the relationships among them, required to develop the model. The chapter also discusses how ANT and Semantic Network are adopted to theoretically and conceptually represent this model.

4.1 Introduction

A context model provides a simplified representation of the real *world* in which the users and devices interact. Hence, a context model provides a systematic way of identifying and representing knowledge about relevant entities and their relationships required to sufficiently represent a situation. Subsequently, a context model forms a basis for representing and reasoning knowledge about situations in a context-aware architecture. Thus, a context model plays a key role on developing a context-aware architecture that intelligently use available information to recognise ongoing situations. Although currently there are various context models, as discussed in section 3.5, they are limited.

This research extends the work in the *theory-based* context models to develop a novel model of situations. This research adopts *Actor-Network Theory (ANT)* and Semantic Network to develop a generic and Knowledge-intensive Context Model (K*i*CM). ANT provides a systematic approach for identifying and representing potential entities and the relationships among them. ANT also treats the entities equally. Therefore, K*i*CM is developed based on a comprehensive list of entities and unbiased relationships. Its entities are described by an extensive list of context parameters and hence makes K*i*CM knowledge intensive. Represented by Semantic Network, K*i*CM is simple and consistently represented.

This chapter discusses how ANT and Semantic Network are adopted to develop K*i*CM. Since design requirements are currently identified in an ad *hoc* manner, section 4.2 synthesises the design requirements for developing context models. Section 4.3 and 4.4 provide an overview of ANT and a discussion on why this

research prefers ANT respectively. Section 4.5 provides a theoretical background of K*i*CM while describing how ANT is applied. Section 4.6 provides a discussion on how Semantic Network notations are adopted to conceptually represent K*i*CM. Section 4.7 illustrates how K*i*CM can be used to model a situation. Section 4.8 provides a summary and conclusion remarks.

4.2 Design Requirements

Researchers have outlined different design requirements when developing their context models. Currently, however, there is no effort to identify, consolidate and formalise design requirements which are generic and applicable to different context models. As noted by Hong *et al.* (2009), such formalisation is important for advancing the research in Context-Awareness. This section describes a set of requirements that should be taken into consideration when developing context models. The rest of this section discusses each of the design requirements.

- **Generic**

Developing a context model is an expensive process because of time, money and resources involved. Therefore, a reusable context model is preferred (Strang & Linnhoff-Popien, 2004). However, different application domains have different situations and hence may require similar entities but different context parameters to represent them. Therefore, a context model should have an extensive but domain independent list of context parameters. In addition, in order to be generic and hence to be reusable, a context model should be *neutral* from any technology. As noted by Newell (1982), a knowledge model should not include

any details on how it should be implemented.

- **Detailed**

Different application domains have different interpretations of situations and thus have different representation requirements of situations. While in one domain, such as healthcare, a user's emotion may be a crucial context parameter, it may not be in other domains. Therefore, in order to facilitate representation of situations in different domains, a context model should include an extensive and domain independent list of context parameters.

Nonetheless, academics in context-aware architectures should not claim to develop a *complete* context model because in reality it is impossible. A model is a simplified representation of a certain reality and therefore a model can never be complete. As noted by Studer *et al.* (1998), *"a model is only an approximation of the reality"*. Kaenampornpan (2009, p. 88) also noted that it is unrealistic to develop a context model by incorporating every aspect of the real world. Therefore a compromise is inevitable when developing a model.

- **Simple**

The principal aim of a context model is to provide a systematic way of identifying and representing knowledge about relevant entities required in order to sufficiently represent a real *world*. Therefore, despite being generic and detailed, a context model should be simple in order to enable designers of context-aware systems to make best use of it. As noted by Lueg (2002), one of the reasons for the users' reluctance from using context-aware systems is system errors. While such errors can be as a result of semantically poor and knowledge limited con-

text models, the complexity of the models can also be a contributing factor. If the model consists of many entities, relationships and context parameters, all meshed up, it can be easily misinterpreted.

- **Consistent**

As noted by Kaenampornpan (2009, p. 69 - 70), unclear relationships between entities lead to inconsistency of interpretations. This leads to context-aware architectures being designed based on misinterpreted context models. In addition, inconsistent interpretation can lead to poor representation of knowledge. Subsequently, as noted by Lueg (2002), the systems become error prone and hence make the users reluctant to use them. Therefore, it is also important that a context model is represented with consistently defined notations and terminologies.

- **Scalable**

Technology is still evolving and therefore developing a scalable context model is important. As new technologies emerge, more sensors may be integrated to context-aware architectures. As noted previously, context modelling is an expensive and time consuming process. Therefore, a context model should be detailed as well as scalable in order to facilitate context-aware architectures to exploit emerging technologies. This requirement is also important in order to enable the model to be adopted in different application domains. Therefore, researchers should not limit their imagination to current technologies.

- **Realistic**

Although a model is implicitly regarded as an abstract representation of a reality, it is worth mentioning that a context model should be realistic. As noted by

Gregory (1993), entities of the model should refer to physical or abstract objects of the real world and the relationships between them should have a logical form as those of the real world. Therefore, a context model should logically specify relationships between entities which are essential.

4.3 Actor-Network Theory

The *Actor-Network Theory (ANT)* emerged in the 1980s as a response to a social and technology divide (Callon 1986; Latour 1991). Unlike typical social theories, ANT focuses on describing the nature of societies without being limited to social relations of human beings. Instead of focusing on frequency, distribution, and homogeneity of human relations, ANT focuses on explaining the influence of technology in societies (Callon 1991, Latour 1992, Law 1992). According to ANT, a society is a set of relationships between heterogeneous human and non-humans, or *actors*, who wish to accomplish a certain task.

ANT defines an actor as any entity that can act or be acted upon (Greimas Courtes 1992; Latour 1996). Unlike typical social theories which inclusively define an actor as a human being, this definition broadens the scope of actors to include non-human actors. To avoid semantic confusions between human and non-human actors, the term actor is replaced by actant. The obvious questions are; (Q_1) What and how actants should be identified in order to define a society? and (Q_2) How should the actants be interrelated in order to define a society?

To address Q_2, the theory defines a network as a point of locus where actants interact. The term *network* is applied in the theory as a point of interaction

of relationships between actants (Latour 1999). Therefore, a network in ANT is used as a metaphor to describe the relationships between actants who wish to accomplish a certain goal. Therefore, the question of how such relationships can be defined in order to define a society is inevitable. To put it in a different way, How can the scope of a society be defined? To answer Q_1 and the follow-up question of Q_2, ANT proposes two principles; *framing* and *disentanglement*. Section 4.3.1 and 4.3.2 provides a discussion of these principles.

4.3.1 Framing

A network is a summing-up of relationships between actants which aim to accomplish a certain goal. According to Callon (1986) and Latour (1999), however, such relationships are dynamically formed as actants interact and are not permanent. The discussion of the permanency of the relationships is provided in section 4.3.2. The theory argues that the relationships between actants should not be stagnant and hence introduce the principle of translation. Later, Callon (1999) refers to this principle as *framing*. Callon (1999) defines framing as a process of identifying distinct actants required to accomplish a certain task. Latour (1999) refers to this process as summing-up.

4.3.2 Disentanglement

Callon (1991) explains the importance of the principle of substitution when identifying actants. This principle is reiterated in Callon (1999) under a different title; *disentanglement*. This principle signifies the importance of actants' flexibility to enter and exit a network. Callon (1991) argues that it is only through

this freedom that networks cannot be caught in a loop. If actants are limited only to a particular network, then there will be a fixed set of actants in each network and therefore only few will exist. This principle is also important for defining attributes, intentions and actions of existing actants. Latour (1999) argues that what matters most in framing actants is their influence on other actants and not what actants can do. Law (1999) argues that actants acquire their attributes when interacting with other actants.

4.4 Why Actor-Network Theory?

This theory is widely applied elsewhere (Kaghan and Bowker 2001; Lamb and Kling 2003; Walsham 2006). This theory, however, has received little attention in Context-Awareness. This raises intriguing question; Why ANT is currently not adopted in Context-Awareness? And why this research prefers ANT over Activity Theory (AT) for modelling situations? The obvious answer to the first question is the emphasis on low level contexts and ad hoc modelling of situations of the majority of the existing solutions. This section, therefore, addresses the second question; Why this research prefers ANT for modelling situations?

4.4.1 It Addresses Similar Questions as in this Research

ANT typically addresses two questions regarding societies while at the same time emphasising the role of non-human actants. The questions include (Q_1) What and how actants should be identified in order to define a society? and (Q_2) How should the actants be interrelated in order to define a society? These

questions have the same goal as one of the problems of this research which investigates *what* and *how* different entities can be utilised to develop a generic and comprehensive model of situations. Among other reasons, the similarities of the emphasis of this theory and the part of the problem which this research is addressing motivate this research to adopt ANT.

4.4.2 Like UbiComp, It Aims to Redefine a User's Life

Weiser (1991) argues that *"the most profound technologies are those that interweave themselves in the fabric of everyday life until they disappear"*. It is evident from the quote that UbiComp places technologies at the centre of the users' everyday life. Likewise, ANT places technical aspects at the centre of the users' everyday life. Therefore, both UbiComp and ANT focus on redefining the users' everyday life. While UbiComp aims to make the users' everyday life better, ANT aims to appreciate the role of non-human actants in the users' everyday life. Unlike UbiComp, however, ANT provides approaches for establishing relationships between human and non-human actants. These approaches can also be adopted to identify relationships between the users and their surroundings and hence make ANT suitable for modelling situations.

4.4.3 It Treats the Potential Entities Equally

In today's computing environments, a presence of one user may affect computing services that are required by another user. Likewise, changes in the states of devices and in the states of physical properties of the environment may have a significant impact on the users and their computing needs. In other words,

each entity in today's computing environments affects other entities. However, Activity Theory (AT), which is adopted in the existing *theory-based* context models, does not support the equality between its entities. AT considers a *subject* as a *super* actant. In contrast, ANT considers its entities equally. In ANT, an actant has a power to enrol and dominate other actants as well as to be enrolled and be dominated. Hence, entities in ANT assume equal roles.

4.4.4 It Takes into Account Dynamic Relationships

Computing environments are *open*, *dynamic* and *heterogeneous* and so the users and other entities within it constantly assume different states and roles. In order for devices to *disappear*, as envisaged by Weiser (1991), there should be negotiations between the users and the other entities. A context model should reflect a certain reality and therefore a modelling approach that takes into account logical forms of relationships between entities as they occur in reality is more preferable. ANT takes into account the dynamism of relationships between actants and therefore it is preferred in this research.

4.5 Theoretical Background of the Model

Section 4.3 and 4.4 addressed the *what* and *why* questions for adopting ANT in this research respectively. However, little has been mentioned about the model and its structure. Situations are dynamic, complex and heterogeneous. Hence, *how* is ANT useful for modelling situations in order to facilitate a design of a context-aware architecture that can reflect the dynamism, complexity and

heterogeneity of situations?

In principle, a situation occurs in a physical environment, henceforth referred to as a *venue*, at a particular time whereby at least one user is involved. In addition, the venue and/or the user has one or more devices. Depending on the category of the venue, more than one situation can simultaneously occur. In common areas such as a Cafeteria, for instance, the users may be chatting with their colleagues while others may be having an informal meeting. While in a *digital world* the users can coexist, in a real world the users can only exist in one venue at a time. Any changes in the venue may imply a change of a situation. A change can be caused by (i) adding new user, (ii) a change of physical properties such as sound, light and temperature, (iii) a change in the state of devices, (iv) a change of the user's physical or psychological states and (v) a change of time.

The *framing* principle emphasises including *relevant* actants when identifying the potential actants. Additionally, the *disentanglement* principle emphasises the impacts the actants have on each other. This principle also emphasises the flexibility of actants to join and quit a network. In addition, the theory defines a network as a point of converging relationships between actants. Since a situation describes the relationships between the users, venue, devices and time, it can be regarded as a network. Subsequently, the users, venue, devices and time can be regarded as actants. However, are these the only *relevant* actants? How do the actants affect each other and what freedom do they have in joining and quitting different situations? The rest of this section addresses these questions.

4.5.1 Potential Entities

Although context is interpreted differently, the literature maintains a consistent list of key entities in the area. The synthesised taxonomy, section 2.5.2, for instance, identifies people, environments, computing resources and time as the key entities. The ontology-based context models, section 3.3.2, also emphasise users, environment, time and computing entities. Thus, the users, venue, time and computing resources, in particular computing devices, are the potential entities for developing a model of situations in this research. As will be discussed in this section, computing service, which is one of the computing resources, is also important and hence this research adopts it as another potential entity.

"People are a major part of the dynamics of work environments" (Schilit, 1995, p. 66). People belong to a certain community and hence their activities are highly influenced by each other. As the majority of devices become mobile, the users' computing needs may also change due to, for instance, social relationships, sensitiveness of information, and emotions of the users. The activity of a research student, for instance, may change as the student's supervisor enters the student's research room. This, subsequently, may change computing services the student and the supervisors may need. Thus the knowledge about the users should also be taken into account when modelling situations.

The users exist in a physical world. Hence knowledge about different venues which are accessible to the users in their daily routines is also important for modelling situations. Apart from location identity, which can be name or number, other physical properties such as temperature, sound and light are also im-

portant. Any change within a venue, which includes any environmental changes such as temperature, sound and light, can imply a change in a situation. In an office, for instance, if the users were quiet and then start talking it may imply that a situation has changed from the users being *'busy working'* to *'working'*.

Knowledge about devices which the users have or are available in different venues is also important for modelling situations. If a meeting, for instance, is strictly known to use a projector, then an absence of the projector in any room means that a meeting situation cannot occur in such rooms. Devices can also be used to determine the users' computer-related activities which are important for recognising the users' ongoing situations. In addition, devices within a particular venue can be a source of sound, temperature and light which, as pointed out in the preceding paragraph, are essential for modelling situations.

It is plausible to ignore computing services and focus on the relationships between the users, devices and venues. These relationships, however, provide no useful information about the users' computer-related activities and hence it becomes difficult to recognise ongoing situation. Analogous to human natural life, if only relationships between the users and their physical environments are taken into account, it will be difficult to infer the users' intentions. When a user wants to print a document, for instance, he/she interacts with a computer which subsequently interacts with a printer. Therefore, knowledge about computing services is also crucial for modelling situations.

The user's activities occur within a period of time and therefore knowledge about time is also important when modelling situations. Similar entities converge in different timestamps in the course of a day to describe different situations. In or-

der to differentiate and keep records of these interactions and the situations they describe, time is a crucial entity. Additionally, most of the activities are structured and hence they have deadlines which may imply a change in a situation. Therefore, to differentiate between these situations time is required.

4.5.2 Relationships Between the Potential Entities

Disentanglement occurs naturally as the users interact in a venue and exits as soon as they leave. Hence, the relationships between the entities are dynamic and they continuously assume different states and roles. This research summarises these relationships into *active*, *heterogeneous*, and *dynamic and temporary* relationships. The rest of this section provides a discussion of these relationships.

- **Active Relationships**

ANT emphasises relationships that influence other actants. In particular, Callon (1986) insists on actants which are within a proximity. Similar emphasis can be found in Context-Awareness (Weiser 1991; Schilit et al. 1994; Dey 2000). In practice, the users affect devices while devices affect the users. The users, for instance, may change the alerting mode of their mobile phones when attending a meeting. Likewise, a mobile phone may affect the mood of the meeting and subsequently its participants if it rings when the meeting is in progress. Similarly, a change of physical properties of the venue may affect users' computer-related activities and consequently devices within the venue. Therefore, the existence of either part continuously affects other entities.

- **Heterogeneous Relationships**

ANT emphasises actants that are unrelated and separated, referring to the process of enrolling actants as *framing*. The users, devices, computing services and venue can be uniquely identified but yet are *"foreigners"* to each other. While mobile devices are usually associated with their owners, each can uniquely be identified. Similarly, although stationary devices are often associated with a particular venue, each can uniquely be identified. Although the devices are associated with and hence inseparable to either the users or the venues, each can uniquely be identified and hence they can be treated as unrelated.

- **Dynamic and Temporary Relationships**

ANT emphasises relationships which are dynamic and temporary. To avoid actants from being *entangled* and caught in looped relationships, ANT emphasises the freedom of actants to join and quit from different networks. In practice, there is the freedom of actant mobility. The users enter and leave different venues within a building while accomplishing their tasks. In the course of a day, the users, and their devices, interact with different users in different venues who also have different devices. Hence, the *disentanglement* occurs naturally as the users, and devices, meet in different venues and exit as soon as they leave.

The analysis identifies the users, venues, devices, computing services and time as the potential entities for modelling situations. The analysis also shows that the entities affect each other and therefore they constantly assume different roles. Although the theory lacks conceptual representation of the entities and their relationships, it can be used to theoretically represent them as a network. In the theory, a network is defined as a point of convergence of the relationships between actants. Hence, the entities and their relationships can also be represented as a

network where a situation will be the point of convergence.

The design requirements, among others, demand a detailed and a generic context model. In order to fulfil these requirements, the synthesised taxonomy for context parameters is adopted. The taxonomy provides detailed knowledge about different entities which are essential for designing context-aware systems. The taxonomy is developed without being motivated by implementation of any context-aware system. Therefore, the specified context parameters are not limited to any sensing technologies or to a specific application domain.

In Computer Science, ideas cannot be well communicated without a conceptual representation. Hence, a conceptual representation of the model is required. The design requirements demand simplicity, consistency and generality of the model. Thus, the question is, what representation notations are adequate for representing the model clearly and consistently? How can this model be represented to enable developers to take advantage of it regardless of their application domains or sensing technologies they have? Section 4.6 addresses these questions.

4.6 Conceptual Representation of the Model

> *"...when I establish a link of some type between two nodes, I am building up a representation of something..." Woods (1975)*

Indeed, when links are established between the users, environments, computing devices, computing services and time, a knowledge-intensive context model is developed. Since Semantic Network is renowned for knowledge representation,

this research adopts it for conceptual representation of the model. A Semantic Network is a graphical notation for representing knowledge. As noted by Woods (1975, p. 14), the unique feature of Semantic Network is the notion of a *link* which connects individual facts into a total structure. This research exploits this notion to conceptually represent the model.

In a Semantic Network, there are two types of links; *property links* and *relation links* (Sowa, 1991). A property link specifies a connection between a node and an attribute or set of attributes while a relation link specifies a connection between two nodes. According to Woods (1975, p. 35 - 37), if a connection specified by a relation link is between two different nodes, then that relation link is an *assertional link*. If a connection specified by a relation link is between nodes of similar type, then that link is a *structural link*. The assertional links specify associations between two nodes while the structural links provide more details about the same node, such as *is-a* relationships in *ontology-based* context models.

Figure 4.1 provides a conceptual representation of K*i*CM. The entities are represented as nodes while the relationships are represented as arcs. To differentiate the nodes and properties, the circle and oval shapes are used respectively. To differentiate the property links and relation links, dotted and solid lines are used respectively. The model does not specify any classification of entities and therefore only assertional links are used. The model uses context parameters from the synthesised taxonomy, figure 2.1. To differentiate the *primary* and *secondary* context parameters, the dotted and solid oval shapes are used respectively. As noted by Henricksen (2003), a context model should cater for uncertainties and hence each entity of K*i*CM is associated with a certainty level, shown as $<p>$.

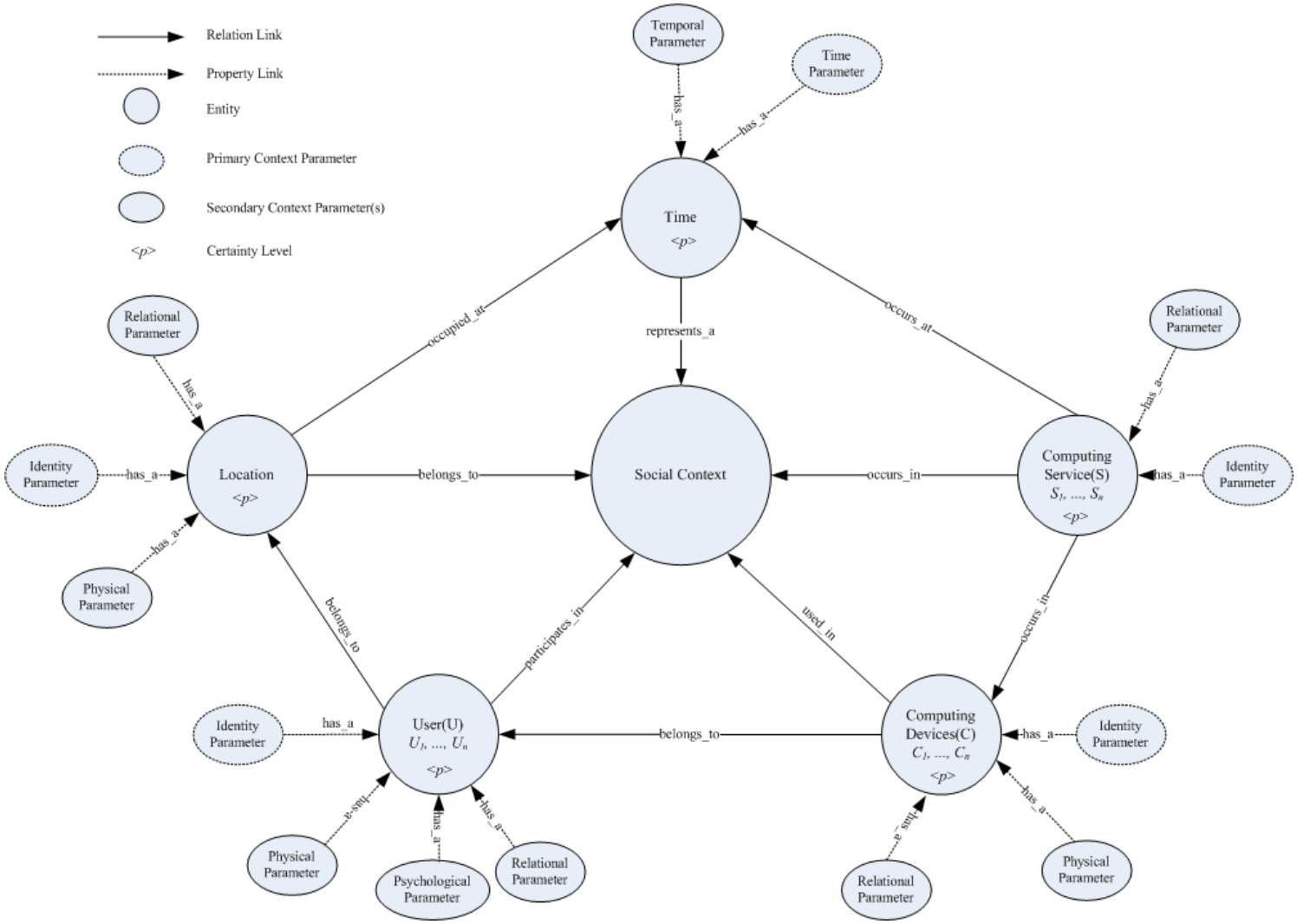


Fig. 4.1: Knowledge-intensive Context Model

According to Woods (1975, p. 24 - 25), each property of an entity should be separately specified by a property link. If an entity has five properties, for instance, then five separate property links should be specified. This rule is useful when a finite property list of an entity can be well defined prior to developing a model. In a case where the property list is ill defined and it changes depending on where the model is applied or on what sensing technology is available, like in Context-Awareness, this rule is inadequate. The links can also be meshed up if the entities specified in a model have many properties. Subsequently, this can make the model too complex and hence difficult to read and interpret.

Thus, to specify the relationships between an entity and its context parameters, this research uses one property link. Since the visibility of an entity is determined by its *primary* context parameter, its relation is separately specified. Hence, if an entity has two categories of secondary context parameters, then three property links are used (two links to specify the relationships between each category of secondary context parameters and the entity, and one link to specify the relationship between the primary context parameter and the entity). As shown in figure 4.1, for instance, the identity of a user is a primary context parameter of the user entity and therefore its relationship to the entity is specified separately from other relationships. So depending on the categories of the property list of an entity, there are at least two specified property links for each entity.

In this model, the primary context parameters of each of the entities are strictly defined as the minimum knowledge required about an entity. As for the time entity, all elements of a timestamp should be included when K*i*CM is used. Through this specification, developers can include all the entities and their re-

lationships but only use a subset of the context parameters. Therefore, instead of instantiating all context parameters of an entity even if only a few are used, as suggested by Kaenampornpan (2009, p. 71 -72), only a subset can be instantiated. Consequently, this enables the model to be reused in different problem domains and to be adjusted accordingly. As is illustrated in section 4.7, K*i*CM can be used to extensively model and represent knowledge about situations.

4.7 A Worked Example of Using K*i*CM

A developer models a situation (i.e identifies and maps knowledge about situations) and then uses the resultant model to represent knowledge about situations in a context-aware architecture. The architecture then uses this knowledge to reason about evidences it gathers from a physical environment to recognise this situation whenever it occurs. K*i*CM supports developers in these two tasks. To illustrate how K*i*CM supports a developer in these tasks, we model a worked example of a situation whereby a research student writes to or reads from his/her computer at his/her desk in his/her research room.

In this section we illustrate how a developer can use K*i*CM to model a situation. The illustration of using K*i*CM to represent knowledge about situations is provided in section 6.2.2. To model a situation using K*i*CM, a developer needs to follow these six steps; 1.) naming of a situation, 2.) identification of instances of the entities, 3.) specifying relationships between instances, 4.) identifying relevant context parameters, 5.) specifying relationships between the instances and their context parameters and 6.) specifying certainty levels of the instances.

The rest of this section describes each of these steps.

- **Step 1: Naming of a Situation**

In this step a developer abstracts a situation with a label that will then be used to identify the situation. In our example, we use **'busy on computer'** label to abstract the situation. In case a developer models more than one situation, then different labels should be used. This is required for differentiating between two or more situations that are within the same problem domain. Logically, the first step should be to identify a situation. In most cases, however, a developer knows the nature of a situation he/she wants to model. Hence including situation identification as one of the steps to model a situation is trivial.

- **Step 2: Identifying Instances of the Entities**

In this step, a developer uses K*i*CM to identify instances of the entities that are required to model the 'busy on computer' situation. In this research an instance means a single occurrence of an entity specified in K*i*CM. To identity instances of the entities, relevant nouns from a description of a situation should be used. In case there is no relevant nouns or instance to match with any entity from K*i*CM, verbs from the description of the situation and domain knowledge can be used to deduce relevant instances.

In the description of our example situation, there are three nouns of interest; a **research student**, a **computer** and a **research room**. These nouns represent instances of the user, the computing devices and the location entities of K*i*CM. To deduce instances of the computing processes entities, we use the verbs *writes* to and *reads* from the description. Since the student writes to and reads from

the computer, then there should be at least two processes to monitor these activities. This research uses Mouse Activity Monitor (**MAM**) and Keyboard Activity Monitor (**KAM**), respectively, to monitor these activities. Section 6.2.3 provides a description of MAM and KAM. Since the student belongs to an organisation, domain knowledge is used to deduce instances of the time entity.

- **Step 3: Specifying Relationships between Instances**

In this step, a developer specifies the relationships between the instances identified in step 2 as indicated on K*i*CM.

- **Step 4: Identifying Relevant Context Parameters**

In this step a developer identifies relevant context parameters of each of the distinct instances identified in step 2. If the identified context parameters are to be gathered from sensors, the developer should make sure that appropriate sensing technologies are in place. If a developer identifies sound, for instance, as an important context parameter for describing a situation then a technology to monitor sound level should be available. Other context parameters are derivable and hence not all context parameters are gathered by sensors.

In our example, **name**, **role**, **officename** and attendance **status** are important context parameters for the user instance. The room **identity** and its **category** are important context parameters for the location instance. The **identity** of the computer, its **owner**, the **room** it is located and its **status** (whether is On or Off) are important context parameters for the device instance. The **name** of the processes, their **host** computer, their **status** (whether are active or inactive) and **timestamps** are important context parameters for the process instances.

Timestamps and their time **categories** (working hours or out of work hours) are important context parameter for the time instance. Table 4.1 provides a summary of the identified context parameters.

Table 4.1: Context Parameters from the 'busy on computer' Situation

Device	Process	User	Room	Time
Identity	Name	Name	Identity	Timestamp
Status	Status	Role	Category	Category
Owner	Host	Office		
Room	Timestamp	Status		

Identity and *name* are primary context parameters since they identify the computer and the room, and the user and the process, respectively. *Timestamp* is also a primary context parameter since it differentiates two points of time. The rest are secondary context parameters. *Office* and *host* of the user and process, respectively, are relational context parameters since they associate the student and the processes with the room and the hosting device respectively. The *owner* and *room* of the device are also relational context parameters since they associate the computer with the student and the room respectively.

- **Step 5: Specifying Relationships between the Instances and their Context Parameters.**

In this step, a developer specifies the relationships between the instances and their context parameters as indicated on K*i*CM.

- **Step 6: Specifying Certainty Levels of the Instances**

Lastly, a developer specifies a certainty level of each of the sensors. A certainty

level is a value that indicates the degree of trustworthy of a sensor. As noted by McKeever (2011), this value can be obtained from training data, domain expert or manufacturer specifications while taking into account users' actions. In this research, five sensors have been used to monitor and gather data about the entities specified in K*i*CM, as illustrated in section 6.2.3. In this example, however, we assume that each of these sensors has a certainty level of 100% i.e 1.0. Figure 4.2 shows the resultant model of the 'busy on computer' situation.

With the attribute-based and ontology-based context models, discussed in section 3.3.1 and 3.3.2 respectively, this situation cannot be modelled as these models do not specify relationship between individual entities. The existing theory-based context models, discussed in section 3.3.3, exclude computing devices and computer-related activities of the users and hence cannot be used to model this situation. The situation abstraction model, discussed in section 3.5, does not specify the entities required to model a situation. This model is also limited to context parameters relevant to a particular task.

Context parameters of each entity is subject to availability of technologies to gather relevant information about that entity. K*i*CM, however, requires the primary context parameters of each entity to be used, at a minimum. This is an important feature of K*i*CM because it ensures that a model of a situation reflects all relevant objects in the real world and their relationships. Nonetheless, the list of secondary context parameters can be as comprehensive as possible so as to model situations close to the reality. This is also an important feature of K*i*CM as it gives developers the flexibility to identify and use parameters that are tailored to their application domain or sensing technologies they have.

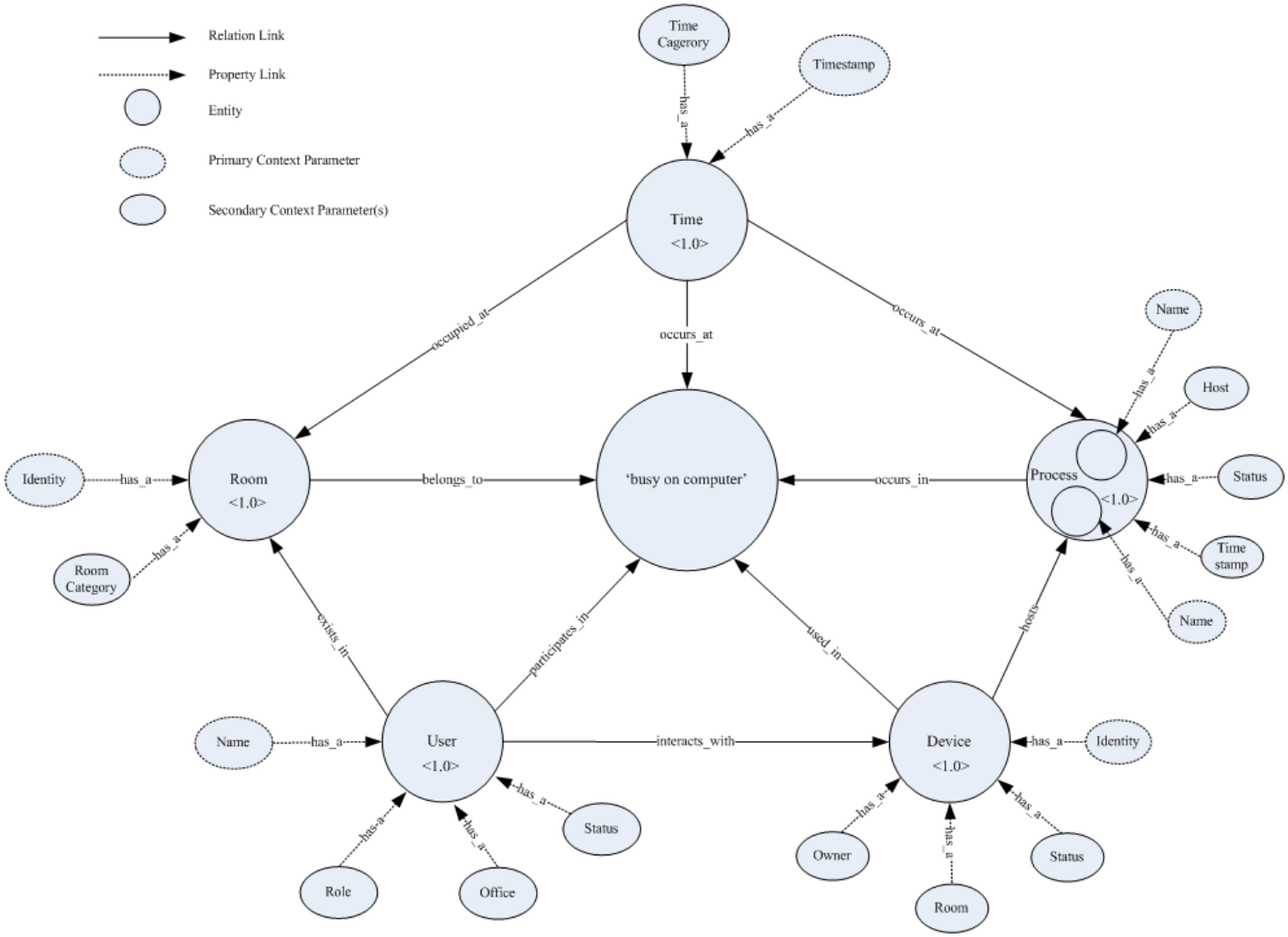


Figure 4.2: A Model of 'busy on computer' by KiCM

4.8 Summary and Conclusion

This research has extended the work in *theory-based* context models by developing a generic and Knowledge-intensive Context Model (K*i*CM). To develop K*i*CM, this research has adopted the Actor-Network Theory (ANT), the synthesised taxonomy and Semantic Networks (SNs). ANT provides a systematic approach to identify and represent potential entities and the relationships among them. This feature enables K*i*CM to be developed by adding computing devices and computing services as among the potential entities for modelling situations.

ANT also treats entities equally and hence each entity in K*i*CM plays equal roles. This is an important feature to K*i*CM as it enables K*i*CM to take into account computer-related activities of nearby users. This makes K*i*CM knowledge intensive and hence more realistic to model situations where the users and their devices continuously interact. Hence K*i*CM can be used to design context-aware architectures that can support more than one user in dynamic environments. The synthesised taxonomy provides an extensive list of context parameters to describe the entities specified by K*i*CM and semantics to differentiate between these parameters.

SNs' notations are used to conceptually represent K*i*CM. SNs are renowned for knowledge representation. These notations provide a clear distinction between nodes, attributes and the type of links required to establish the relationships between nodes, and between a node and its attribute or attribute list. This simplifies and consistently represents the complicated relationships between the entities, and entities and their context parameters in K*i*CM. Consequently, this

representation enables few context parameters to be used to model a situation. This feature gives developers the flexibility to identify and use parameters tailored to their application domain or sensing technologies they have.

Since the entities specified are generic and a subset of context parameters can be used to model a situation, K*i*CM can also be reused. In addition, its generic graphical representation makes K*i*CM simple to use and abstracts it from any knowledge representation formalism. Hence, it gives developers the flexibility of using any knowledge representation formalism and subsequently any inference mechanism. As illustrated in section 6.2.2, knowledge about situations captured by K*i*CM can be represented as a rule and as a Bayesian network.

Chapter 5

Knowledge-driven Distributed Architecture

This chapter discusses *how* a Knowledge-driven Distributed Architecture (KoDA) is designed to take advantage of K*i*CM. The chapter outlines the design requirements for context-aware architecture while taking into consideration the requirements imposed by K*i*CM. The chapter also describes the design of KoDA and discusses each of its components.

5.1 Introduction

Centre to a context-aware system is a context-aware architecture. The architecture provides a mechanism to monitor a physical environment and intelligently use the information to understand the environment and the users' computing needs. This enables context-aware systems to understand their surroundings and the users' computing needs to appropriately respond. To date, there are many context-aware architectures. The existing architectures, however, are designed with little or no consideration of situations. As a result, the architectures use limited or no knowledge about situations and therefore are unable to understand their environments and the users' computing needs.

This research addressed the limitations of the existing context models by developing a generic and Knowledge-intensive Context Model (K*i*CM), in chapter 4. To take advantage of K*i*CM and subsequently to address the limitations of the existing context-aware architectures, this research designs a Knowledge-driven Distributed Architecture (KoDA). K*i*CM influences the design of KoDA by specifying (i) the minimum sensing capabilities (ii) what and how knowledge about entities within a physical environment should be represented and (iii) how knowledge about situations should be represented and reasoned. These design requirements enable KoDA to intelligently use available information to understand its environments and the users' computing needs.

Section 5.2 provides a discussion of the design requirements for context-aware architectures as reported in the literature and as imposed by K*i*CM. Section 5.3 provides a description of the conceptual design of KoDA where each layer and

its components is discussed. Section 5.4 provides a description of the structural representation of the implementation of the architecture. Section 5.5 provides a summary and conclusion remarks.

5.2 Design Requirements

To guide the design of KoDA, a set of design requirements have been derived. These requirements are derived from relevant literature while taking into account K*i*CM. This section provides the discussion of each of the design requirements.

- **Flexible and Scalable**

Technology is advancing rapidly and more technologies are emerging. Therefore, a context-aware architecture should be designed to easily accommodate new technologies. Although currently this is regarded as a technical problem, this research argues that it is also a semantic and knowledge problem. A context model plays a significant role on specifying the entities and context parameters that the architectures can exploit. Therefore, a context model plays a significant role when selecting sensors that can be used. Therefore, the problem should also be seriously considered during developing context models.

- **Distributed Nature**

The majority of computing devices are resource-constrained. As a result, individual computing devices cannot, independently, recognise ongoing situations in order to appropriately respond. Although the research in designing context-aware mobile computing devices are is in progress (Gellersen *et al.*, 2002; Miluzzo,

2011), given the diversity of sensors required to monitor environments, it is unlikely that an individual computing device can sufficiently determine ongoing situations. One of the solutions to this problem is to implement a distributed context-aware architecture. The architecture which utilises a centralised resource-rich computing device for heavy processes which are involved and leave the resource-poor computing devices as the clients.

- **Continuous Monitoring**

It is widely agreed that context is dynamic (Dourish, 2004; Schilit *et al.*, 1994). Therefore the ability of a context-aware architecture to continuously monitor its environment is required. As noted by Dey (2000, p. 29) and Kaenampornpan (2009, p. 80), a context-aware architecture should be able to continuously acquire context parameters. This implies that a context-aware architecture should be able to continuously *listen* to the environment, detect any changes and communicate the changes to the appropriate components of the architecture.

- **Dynamic Inferencing and Responding**

The fundamental goal of a context-aware architecture is to effectively support the users by intuitively providing relevant computing services that the users may require based on ongoing situations. One of the requirements is to continuously monitor the environment. Subsequently, a context-aware architecture should be able to react to any changes that are detected within the environment to dynamically infer ongoing situation and appropriately respond.

5.3 Conceptual Design of KoDA

As proposed by Coutaz *et al.* (2005) and like the majority, this research designs KoDA as a 3-layer architecture but based on K*i*CM. As shown in figure 5.1, KoDA consists of *perception layer*, *inference layer* and *application layer*. The *perception layer* monitors an environment while the *inference layer* intelligently uses available information from the environment to recognise ongoing situations. The *application layer* shares the knowledge about ongoing situations with context-aware applications. The rest of this section describes each of these layers. Appendix B provides a process flow of KoDA.

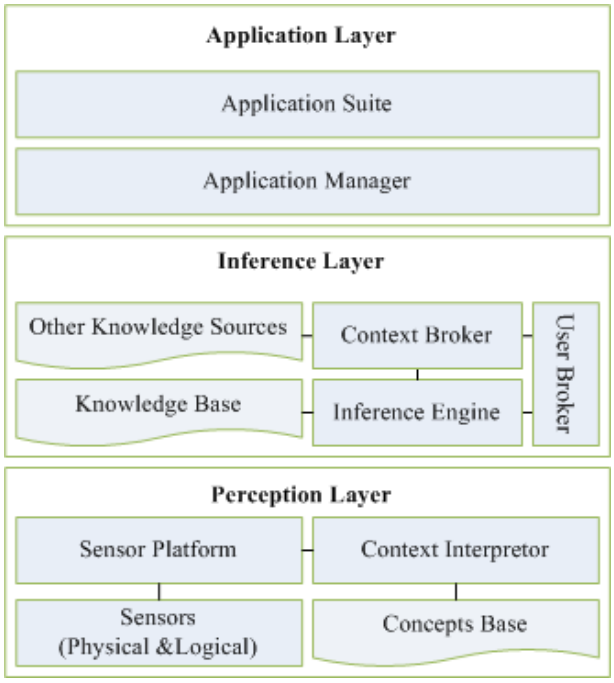


Figure 5.1: Conceptual Design of KoDA

5.3.1 Perception Layer

The perception layer establishes connections with available sensors within an environment, and acquires and interprets data from the sensors. In order to realise these capabilities, the perception layer is designed as an interplay of four components; *sensors*, *sensor platform*, *context interpretor*, and *concept base*. The rest of this section describes each of these components.

- **Sensors**

In order to monitor its environment, KoDA is designed to use various sensors. These sensors can be *physical* or *logical* sensors. Physical sensors are hardware sensors such as Bluetooth and RFID[1] while logical sensors are other sources of data such as applications that monitors network activities or CPU usage. KoDA is designed based on K*i*CM, figure 4.1. Since K*i*CM requires each of its entities to be used, KoDA is designed to monitor all the entities specified in K*i*CM. As is described in the sensor platform, KoDA is designed to easily add new sensing technologies as they emerge. For a discussion on relevant sensing technologies in Context-Awareness refer to Schmidt *et al.* (1999a).

- **Sensor Platform**

In order to accommodate new sensing technologies as they emerge, KoDA is designed with the sensor platform. This platform is a middleware which separates sensors from interpretation processes. This platform is responsible for discovering and establishing connections with and acquiring data from sensors. This platform is designed to contain an array of software modules which are required

[1]RFID is an acronym for Radio Frequency Identification

by different sensors in order to be used by KoDA. This platform, therefore, is a bridge between sensors and a context interpretor. This enables sensors to be added or removed without affecting the rest of the components and hence makes KoDA both flexible and scalable.

- **Context Interpretor**

In order for a context-aware architecture to recognise ongoing situation, it should be aware of different aspects of its environment. In KoDA, this is achieved by monitoring changes within the environments and is accomplished by the sensors and the sensor platform components. Sensors, through the sensor platform, acquire data about different aspects of the environment. The context interpretor is responsible for interpreting this data to context parameters. In addition, the interpretor aggregates and hands over these parameters to the inference engine, which is a component of the inference layer. In order to interpret the data, the interpretor utilises the knowledge stored in a concept base.

- **Concept Base**

In practice, a context-aware architecture is implemented in the real world environment where sensors acquire data specific to that environment. To interpret this data and make use of it, the architecture should have a prior knowledge about entities in the environment. This knowledge is represented by developers using a context model which is designed for that architecture. In KoDA, such knowledge is stored in the concept base and developers use KiCM to represent it. Among others, the concept base stores knowledge about the mapping between the users' true identities and the devices used to identify the users.

As the technology is advancing, it is important for a context-aware architecture to be flexible and scalable in order to allow any changes that may be required to accommodate new technologies. In addition, a context-aware architecture should be able to continuously monitor its environment in order to provide appropriate computing services at all times. KoDA implements the sensor platform which takes care of the sensors and hence it enables new sensors to be added without affecting the rest of the components. The solution to the continuous monitoring of the environment and aggregation of the information, however, is more of a procedure and hence cannot be depicted on the design. As will be demonstrated in chapter 6 the ability of KoDA to continuously monitor its environment is implemented using event-driven programming technique.

5.3.2 Inference Layer

To exploit information, or evidence, collected from an environment to recognise ongoing situation, a context-aware architecture should be able to reason about this information. In KoDA, this is achieved by the inference layer. This layer utilises the information from the perception layer, knowledge about typical situations within an environment, and its reasoning capabilities. To realise these capabilities, the layer is designed as an interplay of five components; *knowledge base*, *inference engine*, *other knowledge sources*, *context broker* and *user broker*. The rest of this section describes each of these components.

- **Knowledge Base**

For a context-aware architecture to exploit evidence collected from an environment, it should possess knowledge about that environment. In KoDA, this

knowledge is represented based on K*i*CM and stored in the knowledge base. The knowledge base is where the rule, the case or the Bayesian network described in section 4.7 is stored. KoDA uses this knowledge and the evidence collected to infer ongoing situation. Hence, the comprehensiveness of a context model is crucial for the richness of knowledge about an environment. Subsequently, this is crucial for the ability of a context-aware architecture to perceive its environment and hence to enable applications to adapt to social dynamics. As discussed in section 3.6, the existing context models are limited and hence their architectures have limited or no knowledge about situations.

- **Inference Engine**

The inference engine is responsible for inferring ongoing situations. Based on evidence collected from an environment, the inference engine assigns truth values to the knowledge stored in the knowledge base based on specified constraints. The engine assigns *true* truth values when the constraints are met and *false* truth values when the constraints are not met. The true truth values mean that a situation that matches the collected evidence is found. In logical-based inference techniques, discussed in section 3.4, the engine terminates the inference cycle if one or more constraints do not match the collected evidence. In contrast, in probabilistic inference techniques, the engine assigns low probability to the recognised situation. In a real world application, this means that the architecture will not provide any computing services to the users or will only provide default computing services which are predefined by developers.

Although providing the users with the default computing services can be the best alternative, it can be a source of the users' annoyance and hence the users'

reluctance to use context-aware applications, as noted by Lueg (2002). To remedy this problem, KoDA is designed to seek other sources of knowledge about situations and to constantly seek the users' confirmation about their ongoing situations. To accomplish this, the inference layer is designed with three additional components; *other knowledge sources*, *context broker* and *user broker*.

- **Other Knowledge Sources**

The other knowledge sources represents any source of knowledge that KoDA can refer to. These sources may include Websites, cooperate Website or mobile devices such as Smartphones and tablets. Through these alternative sources of knowledge, KoDA can access knowledge about, for instance, the users' scheduled events such as meeting, lecture and conference. Unlike the knowledge in the knowledge base, the knowledge from these sources can be in any format. Therefore, a proper mechanism is required to access and render this knowledge to the application layer. In KoDA, this is accomplished by a context broker.

- **Context Broker**

The context broker is responsible for establishing connections with the alternative sources of knowledge, acquiring the knowledge and representing it to the application layer. To accomplish this, the context broker refers to the knowledge about the users and the environment which is specified in the concept base. The context broker, for instance, may acquire calendar entries from the users' Smartphone and transform the output into eXtensible Markup Language which can be interpreted by the application manager of the application layer. In case there is no connection established between the broker and the sources, or no knowledge is acquired from any of the sources, the broker notifies the user broker.

- **User Broker**

The user broker liaises communications between the inference engine and the users. KoDA is designed to request for confirmation of the users' ongoing situation before invoking the required applications. Additionally, KoDA is designed to request the users to specify their ongoing situations in case there is no knowledge about a specific situation in the knowledge base and from the alternative sources. The user broker listens from the inference engine and the context broker for any request of feedback from the users. The user broker sends any feedback to the application manager. Depending on the feedback, the application manager may provide the user with default applications or applications that are appropriate to the ongoing situation.

There are a number of studies that have been conducted to evaluate the usability of context-aware applications and their results show that users of these application feel not in control of their life (Barkhuus & Dey, 2003). This is particularly true in some application domains. In hospital environments, for instance, the users want control of computing services and information they access. In such domains, user feedback is required. KoDA is designed to support the users by providing feedback mechanism.

Although the idea of constantly requesting users' feedback contradicts with the design principles of UbiComp systems, as they are supposed to be *invisible*, it ensures that computing services are offered as the users expect. Subsequently, this makes the users feel in control. However, a trade-off should be made in order to minimise the times when the users need to give feedback. Developers should decide an acceptable balance between *annoying* the users and compromising the

design. In KoDA, this is achieved by inferring ongoing situations by utilising available information within an environment.

5.3.3 Application Layer

The central goal of KoDA is to use available information within an environment to recognise ongoing situation and subsequently enable applications to respond appropriately. Like the majority of the existing context-aware architectures, KoDA is designed with an application layer. This layer is responsible for invoking applications based on the users' ongoing situations. To achieve this goal, this layer is designed as an interplay of two components; application manager and application suite. The rest of this section describes these components.

- **Application Manager**

The application manager is responsible for executing appropriate applications depending on a recognised situation, as inferred by KoDA or as specified by the users. In case no situation is recognised and there is no knowledge from the alternative sources and feedback from the users, the manager will invoke default applications as designated by developers. If the recognised situation is a formal meeting, for instance, the manager searches for appropriate applications and executes them. The typical inputs the manager accepts include a recognised situation and the details of available devices in the room.

- **Application Suite**

The application suite is where the available applications exist. Applications can exist independently or as a suite of applications for a specific function. KoDA

is designed to execute applications from a user's devices and a specialised application server. KoDA, for instance, can execute an application in the a user's Smartphone to remotely change the alerting mode from ringing to silence. Likewise, KoDA can execute an application from an application server to automatically power ON a projector within a particular venue.

The separation of the application manager and the application suite enables either part to be modified without affecting the other. A new application, for instance, can be added in KoDA without affecting how the application manager operates. Likewise, the application manager can be modified without affecting the operations of the applications. This design philosophy enables KoDA to be easily modified and hence more applications can be added as the needs arise. This, as a result, makes KoDA flexible and scalable.

5.4 Structural Representation of KoDA Implementation

Although all the components of KoDA can be implemented in a centralised resource-rich computer, henceforth referred to as a *server*, this research designs some of components to be separately implemented. As shown in figure 5.2, the components of the perception layer are designed to be partially implemented in a distributed computer designated for each venue, henceforth referred to as a *proxy computer*. All sensors for monitoring a particular venue are connected to a designated proxy computer. Except for the interpretation of the data regarding the users, the interpretation of other data is conducted in the proxy computer.

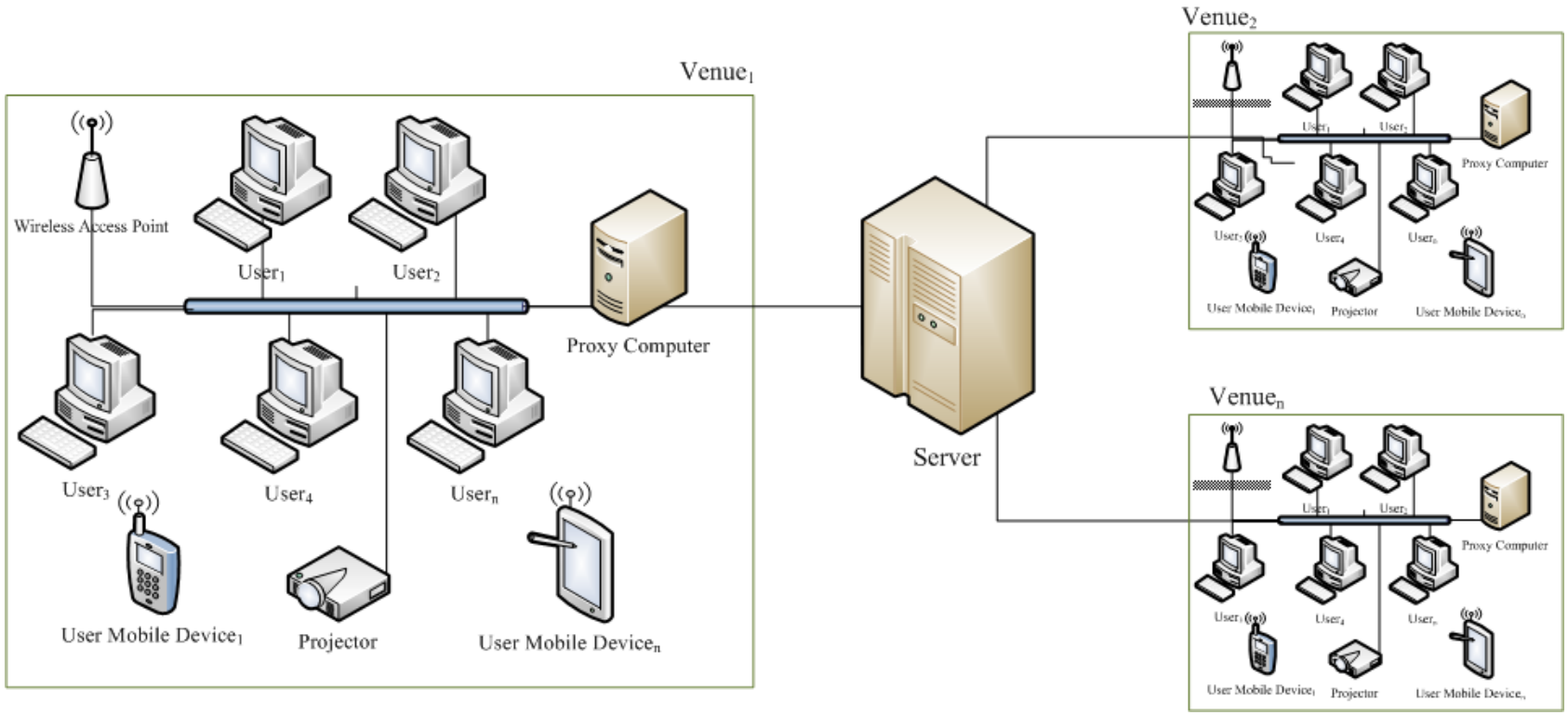


Figure 5.2: Structural Representation of KoDA Implementation

The aim is to separate data acquisition and interpretation processes, which can be resource demanding as the number of sensors increases, from the reasoning processes which by themselves are complex. Instead of the server allocating huge resources to discover and establish connections with each sensor, and acquire and interpret its data, in this design the server only deals with connections with proxy computers and interpretations of their data. This not only fulfils the distributed nature of design requirement and provides a clean design but it also enables KoDA to accommodate a large number of environments. Subsequently, this gives the ability of KoDA to monitor a large part of a building and hence to provide the users with computing services almost everywhere.

5.5 Summary and Conclusion

The analysis of the state-of-the-art revealed that the existing context-aware architectures are developed based on limited context models. These models are inadequate for developing a model of the real world in which the users and devices interact. As a result, these architectures have limited monitoring and reasoning capabilities to use available information to recognise ongoing situations. In chapter 4, this research has developed a Knowledge-intensive Context Model (K*i*CM). This chapter described the design of the Knowledge-driven Distributed Architecture (KoDA) that takes advantage of K*i*CM.

KoDA differs from the existing architectures as it also reasons about information it gathers to recognise ongoing situations. These situations are not confined to a particular task and hence KoDA enables context-aware systems to respond to

social dynamics. Central to KoDA is K*i*CM. K*i*CM facilitates sufficient representation of knowledge about situations in KoDA. Consequently, it enables KoDA to reason about available information to recognise ongoing situations. K*i*CM treats each entity equally and hence knowledge about situations is represented without favouring a particular entity. This enables KoDA to equally treat the users and hence to support one or more users. K*i*CM also imposes minimum monitoring capabilities on KoDA as it requires at least one context parameter of each entity to be monitored.

Complemented by its ability to refer to external sources of knowledge, KoDA provides more flexibility on recognising ongoing situation. Through its distributed nature, KoDA can be used to support multiple rooms within a building. In addition, KoDA can be used to support resource-constrained devices. Through its ability to continuously monitor, infer and respond, KoDA can dynamically recognise ongoing situations. Complemented by K*i*CM, which is developed by generic entities and allows a subset of context parameters to be implemented, KoDA can accommodate new technologies as they emerge. Chapter 6 discusses how KoDA can be implemented.

Chapter 6

KoDA Prototype

This chapter illustrates the application of KoDA on supporting context-aware systems. In this chapter we illustrate how K*i*CM can be used to represent knowledge about situations in KoDA. In this chapter we also illustrate how KoDA can monitor a physical environment, interpret the acquired data and use the resultant information to recognise ongoing situations and to automatically invoke required context-aware systems.

6.1 Introduction

A context-aware architecture plays an important role enabling context-aware systems to respond appropriately to the users' computing needs and their environments. Central to a context-aware architecture is a context model. A context model provides a simplified representation of the real world in which the users and devices interact. Such representation provides a systematic way of identifying and representing knowledge about situations. Although researchers have proposed a number of context-aware architectures, these architectures are unable to reason about information they gather from the environments.

To address these limitations, this research has designed the Knowledge-intensive Context Model (K*i*CM), in chapter 4, and the Knowledge-driven Distributed Architecture (KoDA), in chapter 5. KoDA is designed based on K*i*CM, which is a comprehensive model of the *real world* in which the users and devices interact. This chapter illustrates the use of KoDA and K*i*CM by implementing a prototype. This research uses both an operational context-aware system and a scenario to illustrate the application of KoDA in the real world environment.

Section 6.2.1 and 6.2.2 describe how knowledge about situations was acquired and represented in the prototype respectively. The ability of KoDA to monitor and interpret the data acquired by sensors is illustrated in section 6.2.3 and 6.2.4 respectively. Section 6.2.5 describes how the reasoning capability is implemented and used to recognise ongoing situations. The process of adding and invoking context-aware systems is illustrated in section 6.3. Section 6.4 describes how the prototype was tested and section 6.5 illustrates how KoDA can be used in the

real world environment. Section 6.6 explains how the design requirements are met. Section 6.7 provides the summary and conclusion remarks.

6.2 Prototype Implementation

This research implements a prototype in order to illustrate the feasibility of KoDA. Chen (2004) implements a prototype smart meeting room to support a research group's meetings at the University of Maryland[1]. Likewise, this prototype is implemented to recognise research-related situations of a specific group of research students in a research room. These students are part of the Applied Intelligence Research Centre[2] (AIRC) of the School of Computing at the Dublin Institute of Technology.

Central to KoDA is the generic Knowledge-intensive Context Model (K*i*CM). Section 4.7 illustrated how K*i*CM can be used to model a situation. This section illustrates how K*i*CM can be used to represent knowledge about situations in KoDA. This section also illustrates how KoDA uses this knowledge to dynamically recognise ongoing situations. The pseudo-code, in figure 6.1, represents an algorithm that the prototype implements. The rest of this section describes how the layers of KoDA, and some of their components, are implemented.

[1] http://ebiquity.umbc.edu/
[2] http://www.comp.dit.ie/aigroup/

```
LISTEN Client_Input
REPEAT
    Interprate Client_Input
    Initialise Reasoning_Engine
    Load Knowledge_Base
    Read System_Time
    Open User_Activity_Logger
    Open Inference_Activity_Logger

    IF  Incoming_User IS NOT IN the Room
        Add Incoming_User to List_of_Existing_Users

        IF room IS Office AND Incoming_User IS Member_Of_the_Room
        Switch ON User's Computer

        ENDIF
    ENDIF

    IF  List_of_Existing_Users IS > 1 AND Existing_Users is Member_Of_the_Room
                                      AND Existing_Users NOT Incoming_User
        REPEAT
            Check Keyboard_Activities of Existing_Users
            Check Mouse_Activities of Existing_Users
            Update Status_of_Existing_Users(Username, Keyboard_Status, Mouse_Status)
            Write to User_Activity_Logger(System_Time, Username, Keyboard_Status, Mouse_Status)

        UNTIL List_of_Existing_Users IS NULL

    ELSE
        Set the Status_of_Existing_Users to NULL

    ENDIF
    Insert INTO Reasoning_Engine (Incoming_User, System_Time, Room, List_of_Existing_Users,
                                  Status_of_Existing_Users)

    Fire Matched_Rule
    Write to Inference_Activity_Logger

UNTIL Client_Input IS NULL
```

Figure 6.1: Pseudocode of the Prototype System

6.2.1 Knowledge Acquisition

To acquire knowledge about situations different techniques have been used. Chen (2004), for instance, uses his experience, Kofod-Petersen (2007) uses observation and Kaenampornpan (2009) uses scenarios. This research, like Chen (2004), uses the researcher's experience. Being part of this group, the researcher has observed different situations that have been occurring daily. The researcher has also been regularly interacting with his supervisors and observed numerous similar interactions from other research students. Using this experience, the researcher has outlined six common situations, appendix C. These situations were communicated to and agreed by the other research students of this group.

Table 6.1: Context Parameters Used in this Prototype

User	Device	Computing Service	Room	Time
Name	Identity	Name	Name	Timestamp
Role	Status	Status	Category	Category
Office	Owner	Host		
Status	Room	Timestamp		
Social relation	Category			

These situations were modelled using K*i*CM, as illustrated in section 4.7. This is done by identifying and mapping relevant knowledge about situations to K*i*CM. After modelling these situations, we end up with eighteen context parameters as shown in table 6.1. These parameters and the models form a basis for representing knowledge about situations, monitoring the environment and recognising ongoing situations. Section 6.2.2 describes how knowledge about individual entities specified in K*i*CM and knowledge of these situations is represented.

6.2.2 Knowledge Representation

To represent knowledge about individual entities specified in K*i*CM, eXtensible Markup Language[1] (XML) is used. Alternatively, Ontology can be used but is more useful when software modules from different vendors, and which have different semantics, interact. For software modules developed with similar semantics, as in this research, XML is suitable and hence preferred in this research. The XML document is stored in the concept base of the perception layer of KoDA. As shown in figure 6.2, among others, the document contains knowledge about the users.

```
<?xml version="1.0"?>
<architecture_demo>

    <user role="Student">
        <BluetoothID>null</BluetoothID>
        <TagID>xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx</TagID>
        <name>Student-1</name>
        <role>Student</role>
        <relatesTo>Supervisor-1</relatesTo>
        <office>A-109</office>
        <IP>147.252.229.xx</IP>
        <MAC>xx:xx:xx:xx:xx:xx</MAC>
    </user>
```

Figure 6.2: Excerpt from the XML document

Knowledge about the situations identified in section 6.2.1 need be represented, or *encoded*, in KoDA as inference rules in order to be processed. In this prototype, the rule-based knowledge representation language and the Bayesian network has been used. Our intention is to illustrate the use of both, but separately, logical-

[1] http://www.w3schools.com/xml/xml_whatis.asp

based and probabilistic inference mechanisms and in particular rule-based and Bayesian inference mechanisms, discussed in section 3.4. We used rule-based language and Bayesian network as they are common in Context-Awareness and there are many existing inference mechanisms that support them.

- **Representing Situations as a Rule**

The rule-based is the knowledge representation language in production systems. A production system is a program that provides pseudo intelligence by emulating cognitive ability of a human being (Davis & King, 1975; Newell, 1973).

```
rule "1.0 Busy on Computer"
 when
    $roomResidents: List()
    $userStatus: List()
    $userIn: User()
    $deviceList: List()
    $time: Time(timeFor != "AfterWorkingHours")
    Room($venue: roomname, roomcategory == "Research Room")
    $user: ArrayList(size == 1) from collect (User(userrole == "Student",
    officename == $venue) from $roomResidents)
    not (User(userrole == "Supervisor" || userrole == "Adviser")
    from $roomResidents)
    Device(username == $user.username, room == $venue,
    type == "Computer", status == "ON") from $deviceList
    UserStatus(username == $user.username,
    pname[0].processname == "keyboard", pname[0].status == "true",
    pname[1].processname == "mouse", pname[1].status in ("false", null))
    from $userStatus

  then
    String situation = drools.getRule().getName().substring(4);
    AppManager applicationManager = new AppManager(situation,
    $roomResidents);
    System.out.println("Time: " + $time.time + ", Venue: " + $venue +
    ", Social Context: " + situation);
end
```

Figure 6.3: Rule Representing a 'busy on computer' Situation

In this language, knowledge about a situation is represented as an IF THEN

rule, as shown in figure 6.3. The context parameters are represented as patterns of conditions at the left hand side of the rule while their relationships are maintained by logical operators. The right hand side of the rule specifies actions to be invoked when the conditions are satisfied. In this example, the rule displays a message and invokes the application manager, described in section 5.3.3.

Each model of the situations is represented as a rule where the parameters outlined in table 6.1 are used. As shown in figure 6.3, all primary context parameters do not appear in the rule. This is because the primary context parameters have no direct impact on the occurrence of a situation and hence are indirectly used to determine the secondary context parameters. Hence, the number of the parameters per rule is reduced to thirteen. Since each situation takes different values of context parameters, each model resulted to more than one rule. Hence, we end up with forty six rules in the knowledge base. The excerpts of the knowledge base is provided in appendix D.

- **Representing Situations as a Bayesian Network**

A Bayesian Network (BN) is a directed acyclic graph where nodes represent random variables from a problem domain and directed arcs represent causal relationships between the variables (Heckerman, 1998; Jensen, 1996). A node that causes effects is called a *parent* node while the affected node is called a *child* node. When building a BN, one should start by identifying variables of interest, then establish relationships between these variables and finish by quantifying the identified relationships (Korb & Nicholson, 2003). Quantifying relationships means to specify a conditional probability distribution for each node.

In this research the first two steps of building a BN are simplified since the

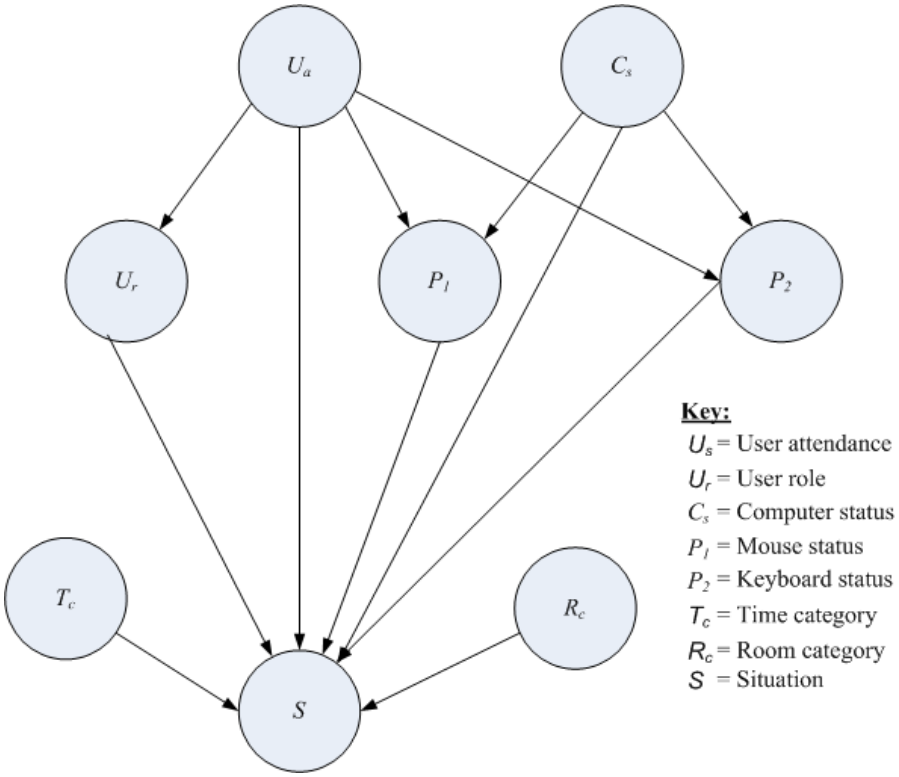


Figure 6.4: Bayesian Network for Situations Involving one User

variables and the relationships between them are specified in a model of a situation. After identifying which variable affects which variable, the resultant BN is shown in figure 6.4. This BN implies that (i) the likelihood of a mouse or a keyboard to be active depends on the user's presence in the room and the status of her computer and (ii) the probability of a situation to occur depends on the presence of the user in the room, her role, category of the room, time and the status of the computer, mouse and keyboard.

The primary context parameters of each of the entities are excluded from this BN as they do not have a direct impact on the occurrence of a situation. The relationships between a device, a user, and a room are implicit and hence are not shown in the BN. A BN allows variables to take different states and hence one

BN has been created to represent the 'busy at desk' and the 'busy on computer' situations, which occur when only one user is in a room. Thus there is no need to show social relation and this reduces the number of parent nodes of this BN to seven. For a BN that involves more than one user refer to appendix E. A Bayesian network can be seen as a knowledge base containing probability clauses. Section 6.2.3 illustrates how KoDA monitors and captures data about relevant entities within the room.

6.2.3 Environment Monitoring

To recognise ongoing situations, the prototype needs information about a physical environment and in particular information about the entities specified in K*i*CM. To gather this information, this research uses both physical sensors and logical sensors. A pair of an RFID[1] reader and antenna has been used to identify the room and the users entered the room. RFID is a promising technology in UbiComp (ODriscoll *et al.*, 2008; Want, 2004). Initially, Bluetooth technology was used but as observed in this research and as noted by Kaenampornpan (2009, p. 254), it is unsuitable for room identification. To monitor whether the users' computers are ON or OFF, a Java function is implemented. To determine time, a Java function that uses the clock of the server is also implemented.

To monitor users' computer-related activities, two applications - Mouse Activity Monitor (MAM) and Keyboard Activity Monitor (KAM) - have been implemented. These applications are installed in the users' computers and are routinely triggered by the server. Both MAM and KAM are written in Java.

[1]RFID is an acronym for Radio Frequency Identification

MAM utilises Java API for mouse monitoring while KAM utilises a Linux utility called logkeys[1]. MAM and KAM continuously listen to and log mouse positions and keystrokes respectively. When triggered, MAM reads its log file and compares the positions of the entries within a specified period of time (e.g 5 minutes). If the positions are different, MAM returns *true* and *false* if otherwise. When triggered, KAM reads its log file to determine if there is any new entry. If there is a new entry, KAM returns *true* and *false* if otherwise.

```
#!/bin/sh

ssh -X $1 java -cp /home/temp/workspace/KBA-Prototype-User/src
                                        MouseActivityMonitor
```

Figure 6.5: A Script for Triggering MAM

MAM and KAM are triggered by the server through shell scripts, as shown in figure 6.5 and 6.6 respectively. The outputs from these applications are transmitted to the server. To establish remote connections between the server and the users' computers, which is mandatory for these applications to be automatically triggered, the Secure Shell cryptographic network protocol is utilised. In particular, this research utilises the OpenSSH[2] tool. This tool is installed and configured in both the server and the users' computers. This tool is configured to allow the server to access the computers without prompting for passwords. Appendices F, G and H provide excerpts from the source code of the server's method for triggering MAM script, MAM and KAM respectively.

[1] http://manpages.ubuntu.com/manpages/maverick/man8/logkeys.8.html#contenttoc2
[2] http://www.openssh.org/

```
#!/bin/sh

ssh $1 java -cp /home/temp/workspace/KBA-Prototype-User/src
                                        KeyboardActivityMonitor
```

Figure 6.6: A Script for Triggering KAM

6.2.4 Data Interpretation

After monitoring the environment, the prototype interprets data from the sensors. Data forms a basis of facts required by the inference engine of the perception layer in order to infer about ongoing situations. Nonetheless, the data captured by the physical sensors is meaningless if it cannot be interpreted in meaningful domain-related concepts, or context parameters. To achieve this, there must be a mechanism to map the data and context parameters of the relevant entities. The knowledge for mapping the data and context parameters is provided in the XML document in the concept base.

To retrieve relevant knowledge from the XML document, the Document Object Model interface[1] has been utilised to implement the context interpretor of the perception layer. Since the knowledge is about a specific entity, Java objects have been created for each entity. Java arrays have also been implemented for each of these objects in order to temporarily store knowledge of different instances of the entities. When the data from a sensor is received, the interpretor retrieves relevant knowledge and creates relevant objects. These objects are then inserted into relevant arrays ready to be inserted into the inference engine. Appendix I provides an excerpt from the source code of accessing information about users.

[1] http://jaxp.java.net/

6.2.5 Knowledge Reasoning

With the use of the rule language, KoDA can now resemble the architecture of a Production System (PS). The PS's architecture consists of production memory, working memory and an interpreter, or a *reasoning engine*, which imitate long-term memory, short-term memory and reasoning capabilities respectively. In this research, the *Knowledge Base* can be regarded as the production memory while the temporary storage of context parameters can be regarded as short-term memory. To implement reasoning capabilities, this research uses the Drools rule engine[1] which implements a *Rete* algorithm (Forgy, 1979).

Rete algorithm is an efficient pattern-matching algorithm (Forgy, 1979). In this language, *Rete* algorithm is the most commonly used algorithm for implementing inference mechanisms. Alternative algorithms include TREAT (Miranker, 1987) and LEAPS (Don, 1994). *Rete* algorithm is preferable in this research because a number of researchers in Context-Awareness have successfully used it, as discussed in section 3.4. Alternative Java-based rule engines that implement *Rete* algorithm include Zilonis[2] and Jess[3]. The Drools rule engine is preferred in this research because it offers an explanation facility, which is particularly important in this research for performance evaluation of KoDA.

The algorithm outlines procedures required to match patterns with the observed facts while the reasoning engine implements the procedures. These procedures are *match*, *select* and *act*. In the *match* procedure, the algorithm compares the

[1] http://docs.jboss.org/drools/release/5.5.0.Final/drools-expert-docs/html/index.html
[2] http://www.zilonis.org/
[3] http://herzberg.ca.sandia.gov/

patterns of each of the existing rules with the set of context parameters. The possible outcome from this procedure is (i) no match, (ii) one match, or (iii) more than one match. The role of the *select* procedure is to analyse the output in order to halt the engine (if there is no match) or to choose a rule that should be executed (Forgy, 1982). The role of the *act* procedure is to execute instructions specified in the consequence part of the rule. In this prototype the instructions include triggering the *application manager* and displaying a feedback message.

6.3 Adding and Invoking Systems

The process of adding a context-aware system to be supported by KoDA is simple. A developer of a context-aware system is simply required to add the name of the system in an application configuration file and specify the situations that the system should be invoked. If a developer wants to add system A to be executed in situation X and Y, for instance, she just needs to open the configuration file and type two lines of texts; A, X and A, Y. A developer is also required to add executable file of her system in the same folder the configuration file is stored. The name of the executable file should similar to the name of the system specified in the configuration file.

To illustrate the process of invoking context-aware system, we use a worked example of an application that automatically switches ON or OFF the users' computers depending on a situation. When a situation is recognised, the *application manager* is invoked and supplied by the name of the recognised situation and the details of the users involved. Using this knowledge, the *application man-*

ager reads the application configuration file and determines that the application to switch computers ON or OFF needs to be invoked. The *application manager* reads information about the users and extracts IP addresses of their computers and uses this information to invoke the application. The source code that shows the implementation of these processes is provided in appendix J.

6.4 Testing of the Prototype

A number of tests were conducted before using the prototype. These tests aimed to identify if the prototype appropriately monitors its environments, interprets data acquired by its sensors and uses the resultant information to recognise ongoing situation. These tests were conducted by the researcher of this research and other independent researchers within the research group. The emphasis of these tests is not on assessing how accurately the prototype recognises situations. Evaluation on accuracy of situation recognition is provided in chapter 7.

The results of these tests are appealing. All sensors are 100% accurate. The prototype also appropriately monitors its environments, interprets data acquired by its sensors and uses the resultant information to recognise situations. These tests also raised an important issue about the RFID reader and antenna used in this prototype. This research utilises a mid-range reader and antenna and hence have distance limitations. These tests revealed that the pair is only reliable within a 20-cm range. Section 6.5 provides application scenarios of KoDA.

6.5 Application of KoDA

To evaluate a context-aware architecture, framework or middleware, researchers use imaginary or working context-aware systems. Kaenampornpan (2009), Kofod-Petersen (2007) and Henricksen (2003), for instance, use scenarios to illustrate how their solutions can be used. Biegel (2005), Chen (2004) and Dey (2000) implement and use working context-aware systems to demonstrate how their solutions can be used. Using scenarios is a feasible approach since the focus of these solutions is not on implementing context-aware systems. Nonetheless, scenarios are far from reality to fully show the potentials of these solutions. Hence, this research adopts both approaches to evaluate KoDA. In addition, this research provides a performance evaluation of KoDA, in chapter 7.

6.5.1 An Application to Switch a Computer ON/OFF

This application remotely switches ON or OFF a computer depending on a situation. This application is developed as part of this research. When entering the room where the prototype is implemented, a user points his/her RFID card on the RFID reader. The prototype interprets this data to identify the user and the room entered, and determines the user's role and the room's category. Concurrently, the prototype records the time the user entered the room and determines whether it is during working hours, lunch hours or not. The prototype also identifies the user's computer and checks whether it is ON or OFF. If the computer is OFF, the prototype triggers the application to switch it ON.

Since only one user is in the room and he/she just entered, there are no computer-

related activities to be recorded from the user. The prototype waits for five minutes or for another user to enter the room to check the status of the user's mouse and keyboard activities. The five minutes waiting is to prevent the prototype from recognising situations based on temporary changes, such as when a mouse is used. After five minutes, the prototype checks if the user has been using her computer by checking her mouse and keyboard activities. Using this new knowledge and the previous knowledge about the room, the prototype recognises the ongoing situation. For illustration purposes, after the situation is recognised the prototype invokes the application to switch OFF the user's computer.

Consider that this is an application to remotely change the alert mode of the users' mobile phones. If the recognised situation is a meeting, for instance, KoDA would trigger this application to seamlessly change the settings of the user's phone from a ringing mode to a silence or vibrating mode. The users would not have to worry about where they enter and what settings their phones are. KoDA would make the use of mobile phone intuitive and thus contributing to the vision of UbiComp.

6.5.2 Microsoft Cortana with KoDA

Microsoft Cortana[1], henceforth referred to as Cortana, is an intelligent personal assistant application for Microsoft Smartphones. It combines voice recognition and context-awareness to effortlessly assist a user. One of the boasting feature of Cortana is its ability to automatically transfer phone calls to voicemail when you do not want to be disrupted. This feature is useful, for instance, when you are

[1]http://www.microsoft.com/en-us/mobile/campaign-cortana/

briefing your boss about a product or giving a keynote speech in a conference. With Cortana installed in your Smartphone, you simply need turn it ON when you do not want to be disrupted and OFF when you are in your normal routines.

Nonetheless, Cortana cannot recognise ongoing situations and hence depends on a user to turn the feature ON or OFF. Subsequently, this requires a user to continuously be aware of her social settings and settings of her Smartphone to effectively use this feature. So before meeting your boss for the briefing, you need to remember about the meeting and turn this feature ON beforehand. Once you finish the meeting, you need to remember to turn this feature OFF. Using Cortana with KoDA removes the need of the users to continuously remember about their social settings and the settings of their devices. Hence there is a potential for increasing the users' productivity when KoDA is used.

6.6 Fulfilment of the Design Requirements

Section 5.2 outlines the design requirements for developing context-aware architectures. This section explains how each of these requirements is met. Table 6.2 provides a summary of these requirements.

To implement R_1 and R_2, this research adopts network programming technique in order to facilitate communications between the server and the clients (proxy and a user's computers). The technique exploits network resources to decentralise the prototype's components. This research exploits the technique to separate the components which deals with monitoring and decision making. In addition to freeing the server for resource-intensive processes, the separation

Table 6.2: Summary of the Design Requirements for Architectures

	Requirement	Details
R_1	Flexible and Scalable	An architecture that can be easily modified to accommodate new sensors as they emerge.
R_2	Distributed Nature	An architecture with multiple software and hardware components interacting through a network or several networks.
R_3	Continuous Monitoring	An architecture which can detect any changes whenever they occur.
R_4	Dynamic Inferencing and Responding	An architecture that can infer and dynamically respond to ongoing situation whenever there are changes.

also enables different proxy computers and users' computers located in different rooms to be connected. This makes KoDA capable of supporting different rooms. This, however, has not been illustrated in this prototype but addressed in the designing of KoDA. The proxy computer transmits data regarding the room to the server. Appendix K and L provide excerpts from the source code of communications between the server and the proxy computer respectively.

```
public class ReaderEvent extends EventObject{

    private static final long serialVersionUID = 6259664790364719027L;

    public ReaderEvent(Object source){
        super(source);
    }
}
```

Figure 6.7: Source Code for Reader Event

To implement R_3 and R_4, this research adopts an event-driven programming technique. In this technique, procedures are automatically executed based on predefined actions or events such as users' or sensors' inputs. This technique is

adopted to continuously listen to RFID-chipped ID cards from the *proxy computers* and mouse movements from the users' computers respectively. These features enable the prototype to spontaneously trigger appropriate procedures required to recognise ongoing situations. Figure 6.7 and 6.8 provide source code for an event and event listener for listening to the cards. Appendix M provides excerpts of source code for the handler of reader event.

```
public interface ClientListener {

    public void tagIdReceived(ReaderEvent event);
    public void newTagIdReceived(ReaderEvent event);

}
```

Figure 6.8: Source Codes for Reader Listener

Unlike other prototypes, like *EasyMeeting* (Chen, 2004), this prototype is implemented to automatically gather data about its inference and a user's computer-related activities. This feature is very essential for performance evaluation of the architecture. Unlike the majority, therefore, this research also evaluates performance of the architecture through various experiments as recommended by Weiser (1991) and Tichy (1998). Unlike Kofod-Petersen (2007), however, this research evaluates performance of the architecture as a whole and in the real world with real users. Chapter 7 describes performance evaluation of KoDA.

6.7 Summary and Conclusion

This chapter has illustrated how the Knowledge-driven Distributed Architecture (KoDA) can be used to support context-aware systems. This chapter has illustrated how Knowledge-intensive Context Model (K*i*CM) can be used to represent knowledge about situations in KoDA. In this chapter we have illustrated how knowledge about situations, represented using K*i*CM, can be represented using rule-based knowledge representation language and the Bayesian network.

This chapter has also illustrated how different sensors can be used to implement the perception layer of KoDA. This chapter has also illustrated how the inference layer of KoDA can use the *encoded* knowledge about situation and the information gathered from the environment to recognise ongoing situations. This chapter has also illustrated how context-aware systems can be added and supported by KoDA. In particular, this chapter has described how the application manager of KoDA uses the knowledge about a recognised situation to automatically invoke the required context-aware systems.

This chapter has also described the pilot tests that have been done to the prototype. The results of the tests are appealing but raised an important issue regarding the RFID reader and antenna used in the prototype. The tests revealed that the pair is only reliable within a 20-cm range. This chapter has also explained how the design requirements for context-aware architectures are met. In particular, this chapter has explained how the network programming and the event-driven programming techniques have addressed these requirements. Chapter 7 provides a discussion on the performance evaluation of KoDA.

Chapter 7

Performance Evaluation

In section 1.5, this research hypothesised that if the relations between the users, an environment, time, devices and computing services are specified and utilised, then a context-aware architecture can dynamically and accurately recognise the users' ongoing situations. In chapter 4 a context model that takes into accounts these entities is developed. This model is used in chapter 5 to develop KoDA. This chapter evaluates KoDA by assessing the accuracy on situation recognition.

This chapter shows that when KoDA recognises situations based on limited knowledge of the users, time and location, the accuracy of situation recognition is very low. This accuracy increases significantly when knowledge of the users' computer-related activities is included in situation recognition. This chapter also shows that this accuracy is further increased when knowledge of certainty level of each sensor is included in situation recognition.

7.1 Introduction

To date, researchers evaluate their context-aware architectures by illustrating how they can be used in the real world, as illustrated in section 6.5. Among the researchers, only Kofod-Petersen (2007) evaluates the performance of his architecture by evaluating its parts separately. As noted by Weiser (1991) and Tichy (1998), experiments are crucial in scientific research in Computer Science and are core to the evaluation of any UbiComp system. While evaluating each component of the architectures can be useful, evaluating them as a whole is important in order to draw fair and useful conclusions. Hence, this research also conducts experimental evaluation of KoDA as whole.

Since context-aware architectures are designed to be used in real world environments, evaluating them in a natural setting of the users is very important. Hence, this research evaluates KoDA both in a real world environment and offline. In the real world evaluation, the prototype implemented in chapter 6 is used by real users in their working environment. This research has implemented one application for illustration purpose and hence there is little the users can benefit from it. Thus, apart from the illustration in section 6.5, the usability evaluation of KoDA is not conducted in this research.

Section 7.2 describes the experimental dataset where the description of sensors used in the evaluation is also provided. The methodology of this evaluation, which includes evaluation criteria and statistical significance, is provided in section 7.3. Section 7.4 describes the evaluation of KoDA in the real world environment and provides a discussion of its results. Section 7.5 describes the offline

evaluation of KoDA and provides a discussion of its results. The discussion of the overall results is provided in section 7.6. Section 7.7 provides a summary and conclusion of the evaluations.

7.2 Experimental Dataset

In this research a dataset of 929 records was gathered over ten days by monitoring three research students in their research room. The number of participants in the room varied but was at most three at any particular time. This dataset is available for download at http://www.ahisec.com/research. This dataset includes data gathered by the prototype, as evidences to its inferences, and the recognised situations.

The datasets provided by PlaceLab (Logan *et al.*, 2007), Van Kasteren *et al.* (2008) and McKeever (2011) were considered in this research. These datasets, however, focus on one user and hence are unsuitable for assessing the accuracy of situation recognition when more than one user is involved. Nevertheless, using a dataset from a third party in this evaluation means sensor events similar to those used to gather the dataset should be provided. This requires a compromise to the prototype, which can be error-prone.

7.2.1 Sensors Description

This prototype is implemented to gather data about (i) the users and their mouse and keyboard activities, (ii) the room the users occupy, (iii) the computers the users use while they are in the room, (iv) the time the users occupy the room

and (v) the time the users interact with their computers.

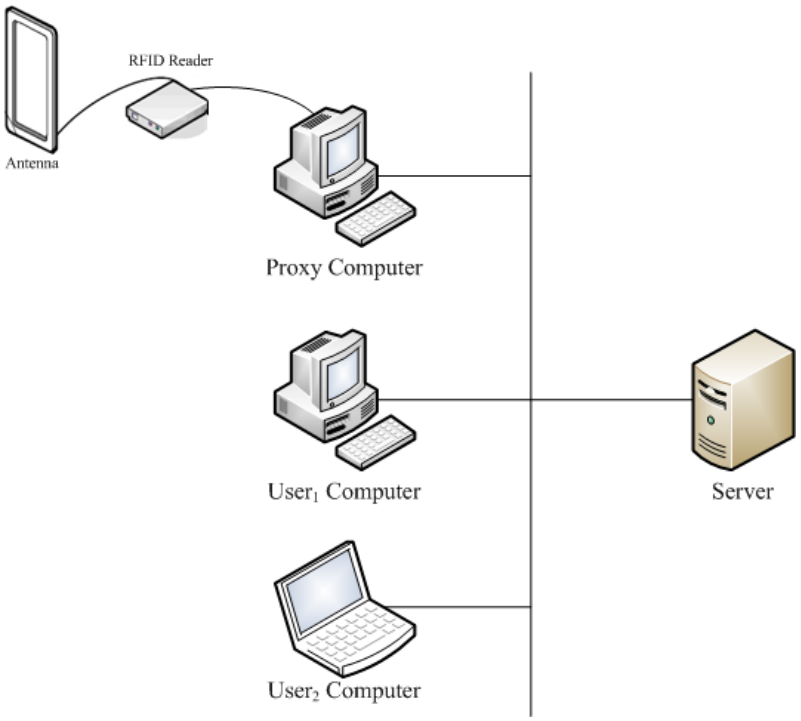


Figure 7.1: Conceptual Representation of the Prototype

To gather this data, this prototype is implemented with five heterogeneous sensors (one physical and four logical), as described in section 6.2.3. A pair of an RFID reader and antenna has been used to identify the room and the participants. As shown in figure 7.1, the antenna is connected to the proxy computer through the reader and the proxy computer is connected to the server through a LAN. To be identified in the room, each participant was given an RFID card and asked to wave the card on the reader when entering or leaving the room. To monitor whether the users' computers are ON or OFF, a Java function, henceforth referred to as a *computer sensor*, has been implemented in the server. To determine time, a Java function, henceforth referred to as a *time sensor*, has

also been implemented in the server.

To gather data about mouse and keyboard activities, each of the participants' computer has been installed with KAM and MAM. Siewiorek *et al.* (2003) utilise ten minutes of sensor data to determine the user's state. This research utilises a five-minute duration, which is commonly used in Operating Systems, to determine if the participants have recently used their keyboard or mouse. The server recognises situations whenever a participant enters or leaves the room, or after every five minutes and hence KAM and MAM are only triggered during this time. This is an important feature as it prevents situations from being recognised based on temporary changes, such as when a mouse is used.

7.2.2 Situations in the Dataset

The participants annotated four situations; (1) busy on computer, (2) busy at desk, (3) working and (4) busy working. Each of these situations is derived from combining thirteen context parameters that are derived from the five sensors, as described in section 6.2.1 and 6.2.2. At any particular time, KoDA is designed to recognise one situation of the user(s).

To annotate their ongoing situations, participants used experience sampling form, shown in appendix N. In this form the participants filled in details such as name of a situation, the time it occurred and other participants who were in the room. Although it is more correct to fill in the forms whenever there is a change in the room, the task was too demanding. Hence, by consulting the participants, a five-minute time slice was determined to be reasonable.

7.2.3 Data Preparation

The ID of the reader and of each of the cards is abstracted to the ID of the room and name of each of the participants, respectively. Using this data, the prototype also determines the participants' role and the room's category. The output from the computer sensor is abstracted to 'on' or 'off'. The output of the time sensor is abstracted to 'WorkingHours', 'LunchHours' or 'AfterWorkingHours'. The outputs of MAM and KAM are abstracted to 'true' or 'false'. The MAM and KAM outputs are 'true' if mouse positions are different and if their is new keystroke in the last five minutes, respectively, and are 'false' if otherwise.

This data was gathered at the same time the data about the actual occurred situations was gathered by the participants. The data about the actual occurred situation is also available for download at `http://www.ahisec.com/research`. Since the prototype was also inferring situations when a participant enters or leaves the room, only instances of recognised situations that correspond to the data about the actual occurred situations were used. Since the prototype recognises situations after every five minutes and the users annotated their ongoing situation after every five minutes, the five-minute time slice is used.

This data was gathered with different settings on the prototype on each day. In day 1, no participants' computers and computer-related activities were included in situation recognition. In day 2, knowledge of the participants computer-related was included in situation recognition. In day 3 and 4, knowledge of the participants' mouse activities and keyboard activities, respectively, was included in situation recognition. In day 5, knowledge of the participants' mouse and

keyboard activities was included in situation recognition. In day 6, 7, 8, 9 and 10, the time duration to determine whether the participants used their computer was set to 1, 2, 3, 4 and 5 minutes respectively.

7.3 Evaluation Methodology

This research conducted two types of evaluations; an *in situ* evaluation and an offline evaluation. The *in situ* evaluation assessed the ability to use available information to accurately recognise ongoing situations in the real world environment. The offline evaluation assessed the ability to recognise situations with uncertainties. Section 7.4 and 7.5 provide more details about the *in situ* and offline evaluations respectively.

7.3.1 Evaluation Criteria

To our knowledge, researchers of the existing context-aware architectures have not conducted experimental evaluations to evaluate their architectures. Hence there are no established criteria for evaluating context-aware architectures. In section 1.5, this research hypothesised that if the relations between the users, an environment, time, computing resources and computing services are specified and utilised, then a context-aware architecture can dynamically and accurately recognise the users' ongoing situations.

Hence, to evaluate KoDA on its ability to recognise ongoing situations this research adopts three widely used statistical classification parameters; *precision*, *recall* and *f-measure*. Precision is the ratio of the times that a situation is

correctly recognised ($S_{recogCorr}$) to the times it is recognised (S_{recog}). Recall is the ratio of the times that a situation is correctly recognised ($S_{recogCorr}$) to the times it appears in the dataset (S_{inst}). F-measure is the weighted mean of precision and recall which is used to measure an accuracy of a test. The precision, recall and f-measure are calculated by equation 7.1, 7.2 and 7.3 respectively.

$$Precision = \frac{S_{recogCorr}}{S_{recog}} \tag{7.1}$$

$$Recall = \frac{S_{recogCorr}}{S_{inst}} \tag{7.2}$$

$$F - measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{7.3}$$

7.3.2 Statistical Significance

To assess the significance of the outcomes of the experiments, this research adopts the *Fisher's test* and the *binomial test*. This research also adopts 0.05 as a significance level. The Fisher's test is used to assess the probability that the differences between the outcomes of different sets of the experiments could be obtained by chance. The binomial test is used to assess the probability that the outcome of a set of the experiments could be obtained by chance.

The binomial test performs a test about the probability of success in a *Bernoulli process*. The Bernoulli process is a sequence of independent, constant probability of success (Lindley & Phillips, 1976). The experiments in this research deter-

mine how many times the prototype correctly or incorrectly recognises ongoing situations. Each outcome of the experiments does not affect the probability of other outcomes. Therefore, each of the experiments is a Bernoulli process.

7.4 *In situ* Evaluation

The aim of this evaluation was to assess KoDA on its ability to use information, or evidence, it gathers from the real world environment to accurately recognise ongoing situations. KoDA is designed based on Knowledge-intensive Context Model (K*i*CM). Hence, KoDA relies on a number of evidence about the participants, the room, time, the participants' computers and their computer-related activities to recognise ongoing situation. In contrast, the closest that the existing architectures can recognise situations is by relying on limited evidence about users, room and time. Hence it is important in this evaluation;

- Ob_1 - To assess the accuracy of situation recognition when knowledge of the participants' computers and their computer-related activities is missing.
- Ob_2 - To assess the accuracy of situation recognition when knowledge of the participants' computers and their computer-related activities is available.

Ob_1 is important for establishing a baseline while Ob_2 for assessing the impact of knowledge about the participants' computers and computer-related activities on the accuracy of situation recognition.

Table 7.1: Set of Experiments in *In situ* Evaluation

	Objective
Experiment 1	To assess the accuracy of situation recognition when knowledge of the participants' computers and computer-related activities is missing.
Experiment 2	To assess the accuracy of situation recognition when knowledge of the participants' computer-related activities is missing.
Experiment 3	To assess the accuracy of situation recognition when knowledge of the participants' computers and computer-related activities is available.
Experiment 4	To assess the effect of time duration used to determine whether the participants are using their computers or not on the accuracy of situation recognition.

To fulfil these objectives, five sets of experiments were conducted. Table 7.1 states objective of each of these experiments. Experiment 1 took into account knowledge of the participants, room and time in situation recognition. Experiment 2, in addition, knowledge of available computers was used in situation recognition. Experiment 3, knowledge of the participants' computer-related activities was also used in situation recognition. Ob_1 is fulfilled by experiment 1 while Ob_2 by experiment 2 and 3. Experiment 4 was conducted to give more insights on the impact of the time duration used to determine whether the participants are using their computers or not on the accuracy of situation recognition.

7.4.1 Experiment Set-up

The prototype is built with inference rules about these situations. Hence, the prototype uses these rules and evidences it gathers from the room at a particular time to recognise an ongoing situation. In this evaluation, all sensors are equally

trusted (with 1.0 certainty level) and hence the evidences are treated as facts. To assess the accuracy of situation recognition, the data about the recognised situations is compared with the data about the actual occurred situations.

This research assumed that if a participant is working on his/her computer, he/she will not spend more than five minutes without using his/her computer's keyboard or mouse. Hence, this research utilises records of the last five minutes to determine if the participants have recently used their computers' keyboard or mouse, as explained in section 7.2. Thus, if KAM or MAM determines that a participant has used her keyboard or mouse in the last five minutes, it was assumed that the participant was working on the computer at that time.

7.4.2 Experiment 1: Recognition without Knowledge of Computers and Activities

In this experiment, situations were recognised while taking into account knowledge of the participants, the room and time. Table 7.2 shows the precision, recall and f-measure from this experiment. Table 7.3 shows a breakdown of situation recognition errors. In table 7.3, the actual occurred situations are shown in the first column, on the the left side, while the recognised situations are shown in the first row, on the top. Each of the remaining rows provide a breakdown of how each situation was recognised. The recall for the accurately recognised situations is shown in bold.

Table 7.2 shows that 'busy on computer' and 'busy working' are not recognised at all. Looking at the confusion matrix, table 7.3, these situations are recognised entirely as 'busy at desk' and 'working' respectively. In these situations,

Table 7.2: Precision, Recall and F-measure without Knowledge of the Participants' Computers and Computer-related Activities.

	Precision	Recall	F-measure
busy at desk	0.41	1.0	0.58
busy on computer	0.0	0.0	0.0
working	0.37	1.0	0.54
busy working	0.0	0.0	0.0

Table 7.3: Confusion Matrix for Situation Recognition without Knowledge of the Participants' Computers and Computer-related Activities.

	busy at desk	busy on computer	working	busy working
busy at desk	**1.0**	0.0	0.0	0.0
busy on computer	1.0	**0.0**	0.0	0.0
working	0.0	0.0	**1.0**	0.0
busy working	0.0	0.0	1.0	**0.0**

the only distinguishing factor is knowledge of the participants' computer-related activities but is excluded from this experiment. Thus, 'busy on computer' and 'busy working' cannot be distinguished from 'busy at desk' and 'working' respectively. The impact of knowledge of the participants' computer-related activities on situation recognition is shown in section 7.4.4.

7.4.3 Experiment-2: Recognition with Knowledge of Computers

In this experiment, knowledge of the status of the participants' computers was also included in situation recognition. Figure 7.2 shows a comparison of the average precision, recall and f-measure for situation recognition with and without knowledge of the participants' computers. As figure 7.2 shows, the average of

precision, recall and f-measure remain the same when knowledge of the participants' computers is included or excluded from situation recognition. This is because, knowledge of the participants' computer-related activities is excluded from this experiment. Hence, like in experiment 1, 'busy on computer' and 'busy working' cannot be distinguished from 'busy at desk' and 'working' respectively.

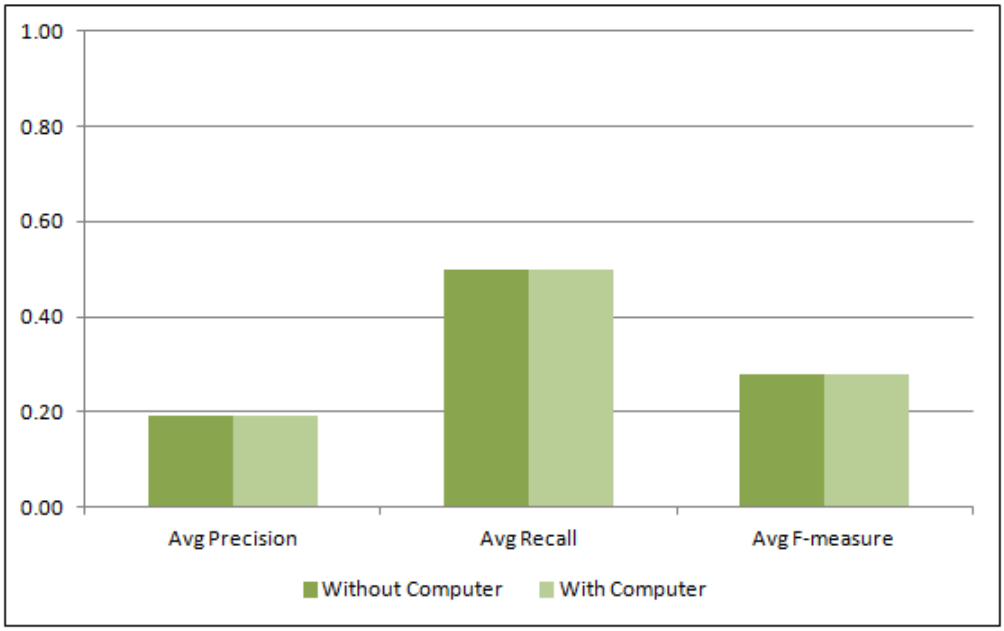


Figure 7.2: Comparison of Average Precision, Recall and F-measure with and without Knowledge the Participants' Computers.

7.4.4 Experiment-3: Recognition with Knowledge of Activities

In this set of experiments, knowledge of the participants' computer-related activities was also included in situation recognition. To assess the effect of this knowledge separately and when combined, this experiment was done in three

steps; (i) when knowledge of mouse activities was used (ii) when knowledge of keyboard activities was used and (iii) when knowledge of keyboard and mouse activities was used. Figure 7.3 and 7.4 show the average precision, recall and f-measure when knowledge of the participants' mouse and keyboard activities was included separately and when combined. Figure 7.4 shows these averages after excluding records when the participants forgot to use their RFID cards.

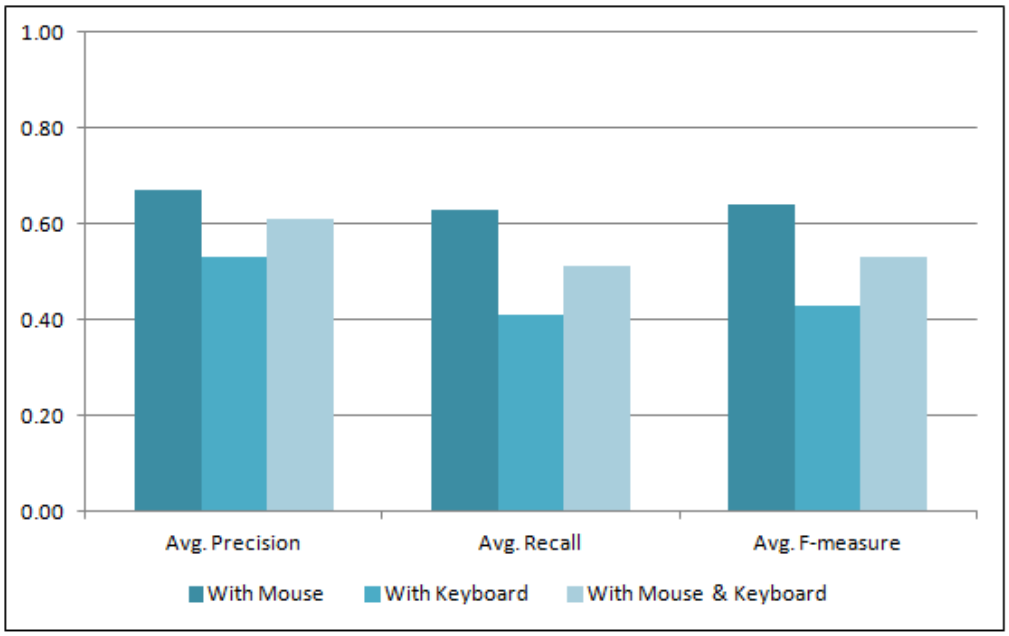


Figure 7.3: Average Precision, Recall and F-measure with Mouse, Keyboard and both Mouse and Keyboard Activities (before excluding records).

As shown in figure 7.4, the average precision, recall and f-measure are almost equal when knowledge of the participants' mouse and keyboard activities is included separately and combined. This is because all sensors were 100% trusted and hence any movement of a mouse or keystroke meant a participant was using her computer. Figure 7.5, 7.6 and 7.7 shows the precision, recall and f-measure,

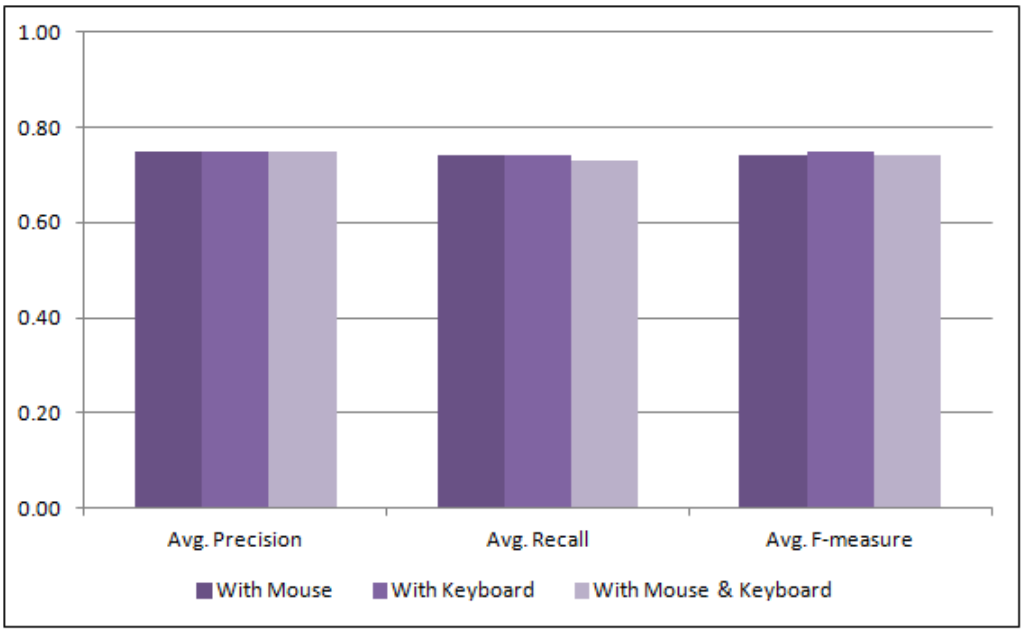


Figure 7.4: Average Precision, Recall and F-measure with Mouse, Keyboard and both Mouse and Keyboard Activities (after excluding records).

respectively, when knowledge of the participants' mouse and keyboard activities is excluded, and when it is included separately and combined.

As shown in figure 7.5, the precision for all situations improves, with that for 'busy on computer' and 'busy working' jumped from 0 to as high as 0.79 and 0.83 respectively. Figure 7.6 shows the recall for 'busy on computer' and 'busy working' also improves, with the exception of 'busy at desk' and 'working'. This is because with the inclusion of knowledge of the participants' mouse and keyboard activities on situation recognition, 'busy on computer' and 'busy working' can now be recognised. Hence, any brief use of a mouse or a keyboard by the participants when were 'busy at desk' and 'working' led to these situations be recognised as 'busy on computer' and 'busy working' respectively.

Figure 7.7 shows that the f-measure for all situations increases, with that for

Figure 7.5: Precision with Mouse and Keyboard Activities.

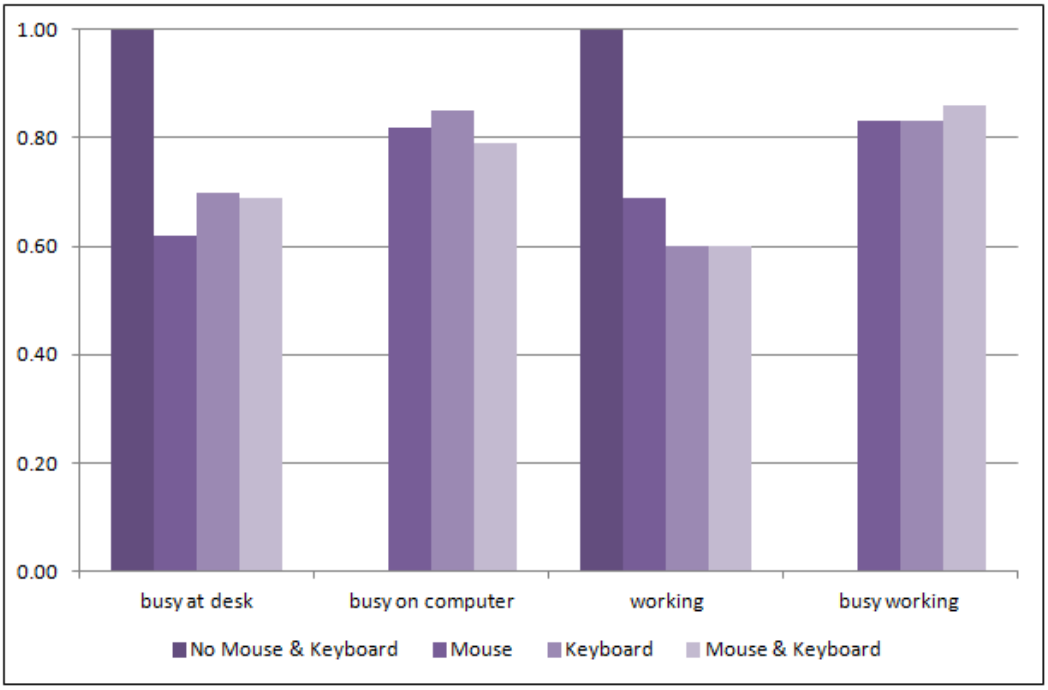


Figure 7.6: Recall with Mouse and Keyboard Activities.

'busy on computer' and 'busy working' jumped from 0 to as high as 0.81 and 0.83 respectively. With inclusion of knowledge of the participants' mouse and keyboard activities in situation recognition, 'busy on computer' and 'busy working' can now be distinguished from 'busy at desk' and 'working', respectively. Looking at the confusion matrix, in table 7.4, 'busy on computer' and 'busy working' are now less confused with 'busy at desk' and 'working', respectively.

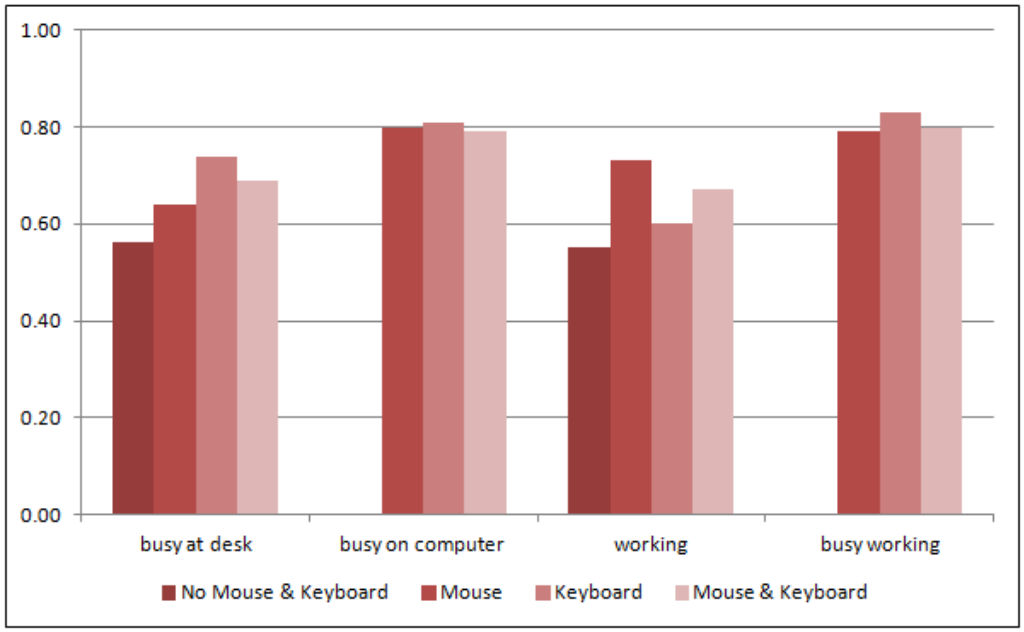


Figure 7.7: F-measure with Mouse and Keyboard Activities.

The confusion matrix also shows 'busy at desk' and 'working' are largely recognised as 'busy on computer' and 'busy working' respectively. This is because the participants sometimes briefly used their mouse or keyboard when were 'busy at desk' or 'working'. Since any mouse or keyboard activity of the last five minutes is valid and situations are recognised after every five minutes, the chances of any brief use of a mouse or keyboard to be included in situation recognition is high.

This suggests that the recall for 'busy at desk' and 'working' will increase as the duration of time to determine whether the participants use their computers or not is decreased. This is shown in experiment-4, section 7.4.5.

Table 7.4: Confusion Matrix for Situation Recognition with Mouse and Keyboard Activities.

	busy at desk	busy on computer	working	busy working
busy at desk	**0.69**	0.31	0.0	0.0
busy on computer	0.21	**0.79**	0.0	0.0
working	0.0	0.0	**0.60**	0.40
busy working	0.0	0.0	0.13	**0.86**

Table 7.5 shows average f-measure when knowledge of the participants' computers and computer-related activities was included and excluded from situation recognition. As table 7.5 shows, the accuracy of situation recognition increases almost three times when knowledge of the participants' computer-related activities is included in situation recognition. To check if this difference is significant, Fisher's test was used as described in section 7.3.2. This test shows that this difference is statistically significant to the 99.9999% level. This table also shows the accuracy of situation recognition is the same if knowledge of the participants' computer is included or excluded from situation recognition.

Table 7.5: Comparison of Average F-measure with and without Mouse and Keyboard Activities.

	No Computers & Activities	Computers	Activities
Avg. f-measure	0.28	0.28	0.74

7.4.5 Experiment-4: Effect of Time Duration to Monitor Activities

In this set of experiments, the effect of duration of time used to determine if a participant is using a computer or not is assessed. This set of experiments was conducted when the duration was set to one, two, three, four and five minutes. As figure 7.8 shows, the average recall increases up to when the duration is three minutes and then decreases. Looking deeper, figure 7.9 shows that the best result for situation recognition is achieved when the duration is three minutes, as opposed to five minutes used in this evaluation.

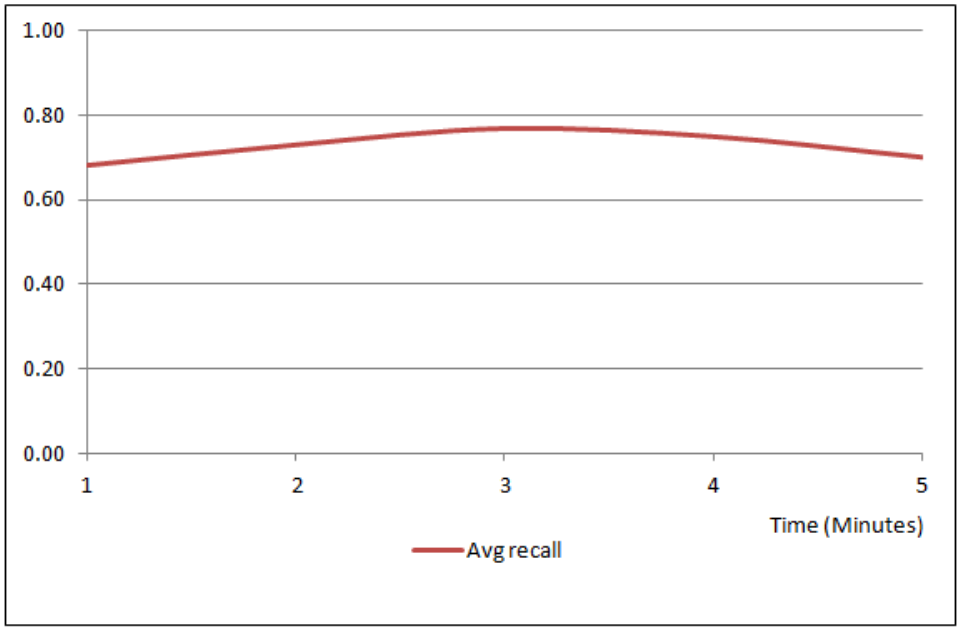


Figure 7.8: The Average Recall with Different Time Durations

As shown on figure 7.9, the average recall for 'busy at desk' and 'working' decreases as the duration increases whereas that of 'busy on computer' and 'busy

working' increases. This is because as the duration to determine whether the participants are using their computers or not is longer, the chances of a brief use of a mouse or keyboard when the participants are 'busy at desk' or 'working' to be taken into account on situation recognition is very high. In contrast, the chances of any duration of time the participants are 'busy on computer' or 'busy working' and not using their mouse or keyboard to be taken into account in situation recognition is very low.

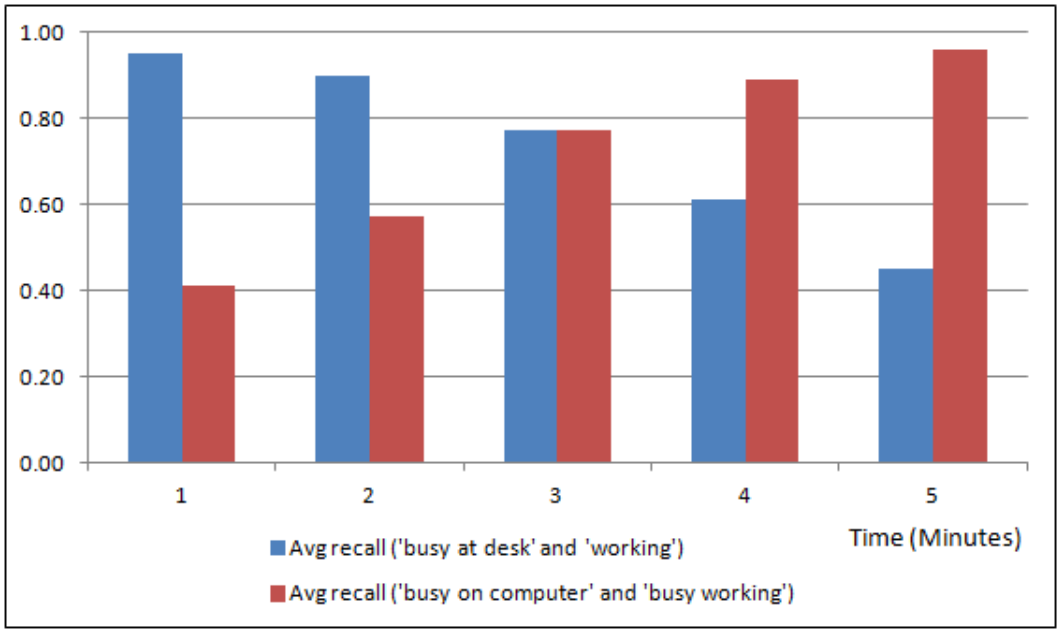


Figure 7.9: The Average F-measure with Different Time Durations

7.4.6 Discussion of the Results

This evaluation shows that when knowledge of the participants' computer-related activities is included in situation recognition, the accuracy of situation recognition improves almost three times compared to when this knowledge is excluded

from situation recognition. This improvement was expected because in the real life, the users are engaged in different computer-related activities in different situations. Hence, by taking into account knowledge of the users' computer-related activities in situation recognition, a context-aware architecture can differentiate between different situations.

This evaluation also shows knowledge of whether the participant's computer is ON or OFF is less important in situation recognition. This finding was also expected because knowledge of whether the participant's computer is ON or OFF, without knowing the participant's computer-related activities, adds nothing to a context-aware architecture to differentiate between different situations. This knowledge is important for determining the participants' computer-related activities. These findings justify our decision to include knowledge of the users, location, time, devices and computer services in our context model.

Further investigation shows the accuracy of situation recognition changes as the duration of time to determine whether the participants interact with their computers changes. This evaluation shows the accuracy of situation recognition improves by 7% when monitoring duration is reduced to three minutes but decreases as the duration is further reduced. Although this improvement is not statistically significant, as shown by the Fisher's test, it gives the impression that the duration of time to determine whether the participants are using their computers or not is an important factor in situation recognition.

The Binomial test shows that the overall probability for accurately recognising 553 or fewer situations in 884 trials is <0.0001. This is statistically significant and hence this research concludes that the results obtained by KoDA are not

Table 7.6: An Overall Probability for Each of the Situations Being Recognised Correctly.

	busy at desk	busy on computer	working	busy working
Probability	0.0537	0.0024	<0.0001	<0.0001

by chance. Table 7.6 shows a breakdown of the probability for recognising each situation. As shown in table 7.6, the probability for accurately recognising 'busy on computer', 'working' and 'busy working' situations is statistically significant except the marginal result of the 'busy at desk' situation.

In this evaluation, all sensors were equally trusted and hence lack of any context parameter resulted in poor performance. As shown in experiment 1 and 2, lack of knowledge of the participants' computer-related activities resulted in 28% accuracy of situation recognition. And as shown in experiment 3, figure 7.3 and 7.4, by excluding records where the participants forgot to use their RFID cards, the accuracy of situation recognition increases. The aim of this evaluation was to assess the ability of KoDA to use information gathered from the real world environment to accurately recognise ongoing situation. Hence, this evaluation did not take into account uncertainties. The investigation of the impact of uncertainties in situation recognition is provided in section 7.5.

7.5 Offline Evaluation

This evaluation aims at assessing the impact of knowledge of certainty levels of sensors on the accuracy of situation recognition. Instead of equally trusting the sensors, as in the *in situ* evaluation, in this evaluation each sensor is assigned

with a certainty level. In this evaluation, situation recognition is done by a Bayesian inference engine instead of the logical-based inference engine.

The dataset described in section 7.2 is used in this evaluation. Since the aim is to assess the impact of knowledge of certainty levels on the accuracy of situation recognition, only records that involve one participant were used. The time duration used to determine whether the participants are using their computer or not is less important in this evaluation. Thus, only records that were gathered when the time duration was set to five minutes were used. Hence, a total of 219 records from the dataset were used in this evaluation.

To accomplish the aim of this evaluation, two experiments were conducted each with one objective;

Ob_1 - To assess the accuracy of situation recognition without knowledge of certainty levels.

Ob_2 - To assess the accuracy of situation recognition with knowledge of certainty levels.

Ob_1 is important for establishing a baseline while Ob_2 for assessing the impact of knowledge of certainty levels on the accuracy of situation recognition.

7.5.1 Experiment Set-up

This research uses the Microsoft Bayesian Network API (MSBN3) to implement the Bayesian inference. The Bayesian network created in section 6.2.2 is used as an input to this inference engine. Certainty levels used in this evaluation

for each sensor is provided in table 7.7. The data from the dataset is used as evidences for situation recognition. To automate the process of acquiring and assigning evidences to the inference engine, this research has developed a tool in C# programming language. This tool also writes inference results to a spreadsheet file and hence simplifies data analysis process. The source code of the tool is available for download at `http://www.ahisec.com/research`.

As described in section 6.4, all sensors used in this prototype are 100% accurate. However, as noted by McKeever (2011), users' actions can degrade quality of information received from these sensors. In this prototype, users' actions are involved on RFID, MAM and KAM. Hence, to determine a certainty level of RFID we used all data from the dataset. We counted the number of times the participants are indicated to be out of the room while KAM and MAM indicated there were mouse and keyboard activities on their computers. We then divided this number by the number of times the participants were required to use their RFID cards. This gives us 0.79 certainty level of RFID sensor.

Table 7.7: Certainty Level of Each Sensor Used in this Prototype

	Time Sensor	Computer Sensor	RFID	KAM	MAM
Accuracy	1.0	1.0	0.79	1.0	0.87

To determine certainty levels of MAM and KAM, we run one test prior to this evaluation. This test ran over three days and involved one user. The user was asked to note her mouse and keyboard activities of one hour in each day while MAM and KAM have been simultaneously logging these activities. The user indicated to use the mouse 41 times while MAM indicated the mouse

has been used 47 times. Thus, 87% of the mouse activities were correct i.e the certainty level of 0.87. Both the participant and KAM indicated that the keyboard was used 28 times. Thus, 100% of the keyboard activities were correct i.e the certainty level of 1.0. Table 7.7 provides the certainty level of each sensor used in the prototype. Appendix O provides an excerpt of probability distribution of parents nodes used in this evaluation.

7.5.2 Experiment 1: Recognition without Certainty

In this set of experiments, knowledge of certainty levels of sensors is excluded from situation recognition. Table 7.8 shows the average of precision, recall and f-measure from these experiments. Table 7.9 shows a breakdown of situation recognition errors. Table 7.8 shows that 'busy on computer' is poorly recognised.

Table 7.8: Average Precision, Recall and F-measure without Certainty.

	Precision	Recall	F-measure
busy at desk	0.61	0.58	0.51
busy on computer	0.51	0.47	0.49

Looking at the confusion matrix, table 7.9, 'busy on computer' is largely recognised as 'busy at desk'. This is because these values are calculated by combining even the situations recognised when knowledge of the participants' computer-related activities is excluded from situation recognition.

Table 7.9: Confusion Matrix for Situation Recognition without Certainty.

	busy at desk	busy on computer
busy at desk	**0.58**	0.42
busy on computer	0.53	**0.47**

7.5.3 Experiment 2: Recognition with Certainty

In this set of experiments, each sensor was assigned with its certainty level as shown in table 7.7. Figure 7.10 shows a comparison of the average precision, recall and f-measure for situation recognition with and without knowledge of certainty levels. As figure 7.10 shows, the average of precision and recall increases, with overall improvement of 22% in f-measure, when a certainty level of each sensor is taken into account in situation recognition.

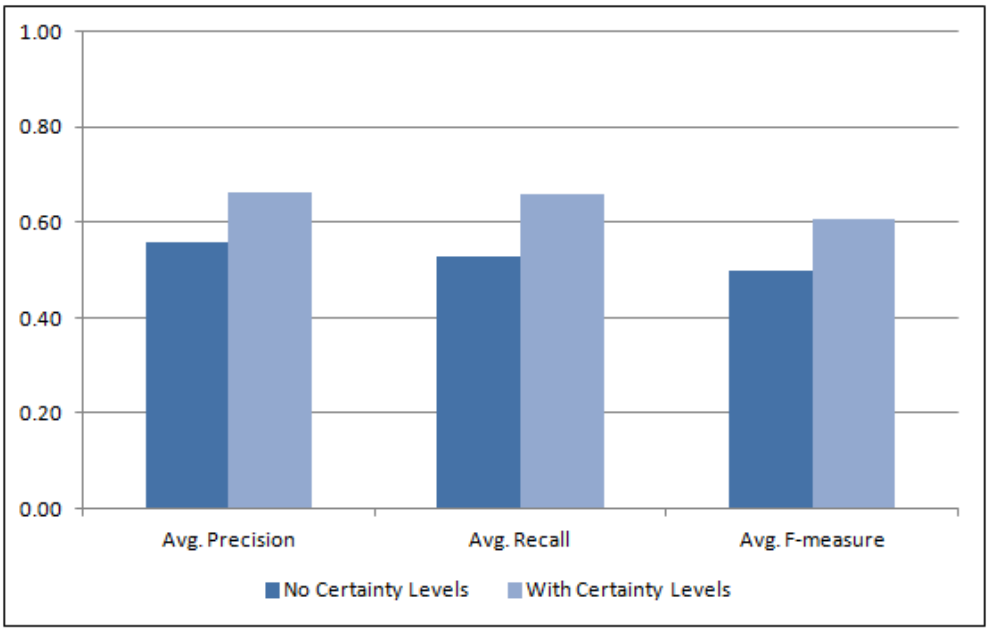


Figure 7.10: The Comparison of Average Precision, Recall and F-measure for Situation Recognition with and without Certainty.

Figures 7.11, 7.12 and 7.13 show a comparison of precision, recall and f-measure by situation, respectively, of situation recognition with and without knowledge of certainty levels. Figure 7.11 shows precision for all situations improves. Figure 7.12 shows the recall of 'busy at desk' decreases while that of 'busy on computer' increases. Subsequently, this causes the f-measure of the 'busy at desk' to decrease, as shown in figure 7.13. This is because when knowledge of the participants' computer-related activities is excluded from situation recognition and the status of their computers is ON, all the 'busy at desk' situations are recognised as 'busy on computer'.

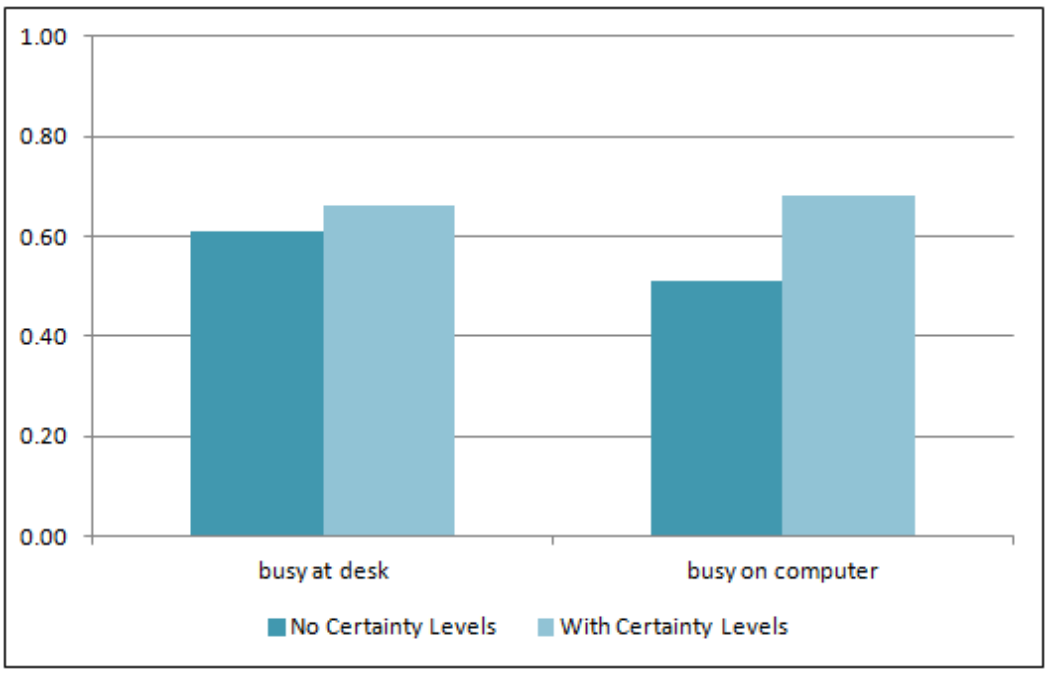


Figure 7.11: The Average Precision with and without Certainty.

Table 7.10 shows average f-measure when knowledge of certainty level of each sensor was included and excluded from situation recognition. As table 7.10

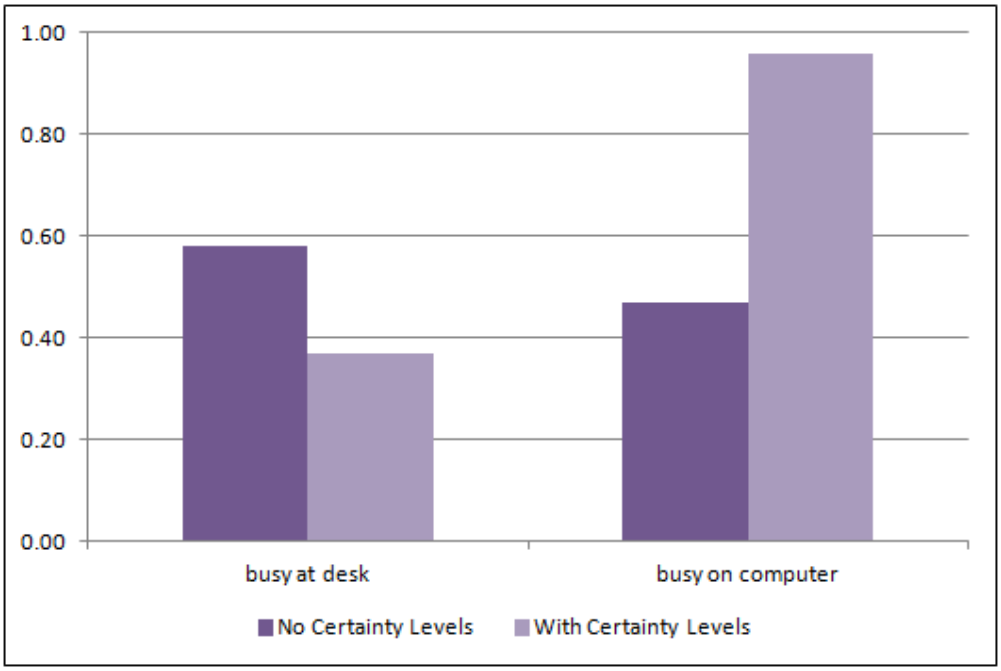


Figure 7.12: The Average Recall with and without Certainty.

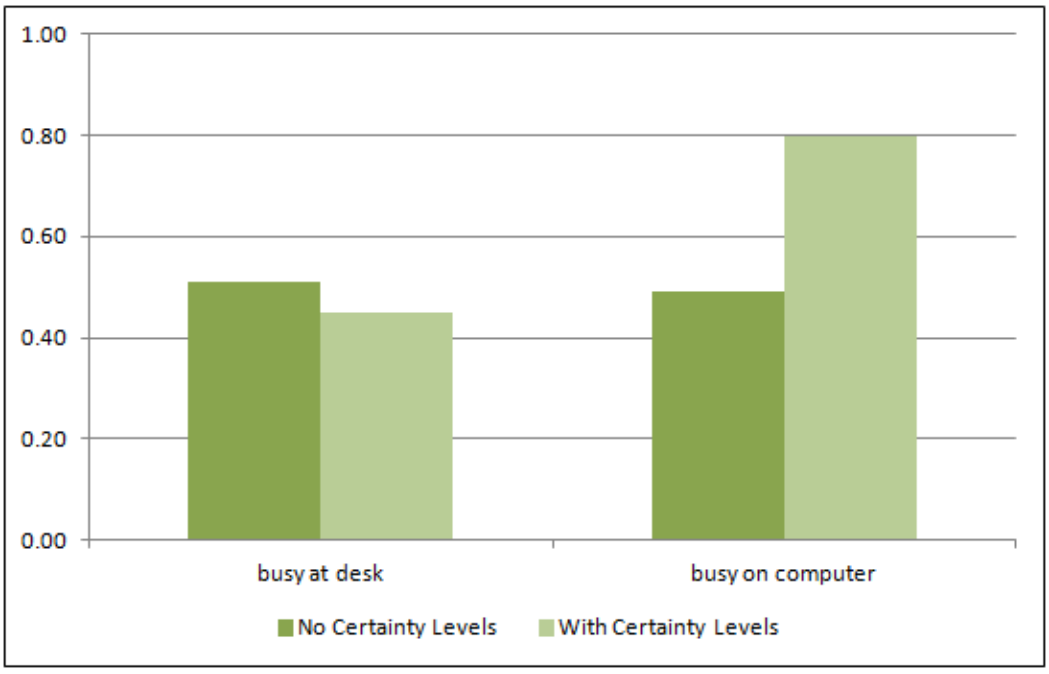


Figure 7.13: The Average F-measure with and without Certainty.

shows, the accuracy of situation recognition improves by 21% when knowledge of certainty levels is included in situation recognition compared to when this knowledge is excluded from situation recognition. To check if this difference is significant, Fisher's test is used. This test shows that this difference is statistically significant to the 99.9999% level.

Table 7.10: Comparison of Average F-measure with and without Certainty.

	No Certainty Levels	With Certainty Levels
Avg. f-measure	0.53	0.66

7.5.4 Discussion of the Results

This evaluation shows the accuracy of situation recognition increases by 25% when knowledge of certainty level of each sensor is included in situation recognition. This improvement was expected because with certainty levels, uncertainties of context parameters are quantified and preserved in situation recognition. Context parameters are treated as beliefs rather than facts and hence situations are recognised even if some of the parameters are missing. With certainty levels, for instance, situations are recognised even in occasions where the participants forgot to use their RFID cards.

7.6 Discussion of the Overall Results

This research hypothesised that if the relations between the users, an environment, time, computing resources and computing services are specified and

utilised, then a context-aware architecture can dynamically and accurately recognise the users' ongoing situations. Our results significantly support this hypothesis. Our results show that when KoDA recognises situations based on limited knowledge of the users, time and location, the accuracy of situation recognition is very low. This accuracy increases significantly when knowledge of the users' computer-related activities is included in situation recognition. Our results also show that this accuracy is further increased when knowledge of certainty level of each sensor is included in situation recognition.

7.6.1 Benefits of KoDA

KoDA makes the use of devices intuitive and hence has the potential to increase the users' productivity. As illustrated in section 6.5, KoDA enables context-aware systems to respond to the users' ongoing situation. It knows when the users' situation changes to dynamically invoke appropriate context-aware systems. Hence, the users do not have to worry about the settings of their devices.

In KoDA, inference rules (or knowledge about situations) are encoded by a developer. This process can be very difficult when there is no model to follow and there are hundreds of sensors and subsequently context parameters to be used. K*i*CM simplifies this process by providing a systematic way of identifying and representing relevant entities, their context parameters and the relationships amongst them prior representing inference rules.

In K*i*CM, each entities has equal importance. This feature enables K*i*CM to take into account knowledge about computer-related activities of nearby users in situation recognition. Hence, K*i*CM can be used to design context-aware

architectures that can support more than one user in dynamic environments. As shown in the evaluations, KoDA takes into account the computer-related activities of all of the existing users in recognising ongoing situation.

K*i*CM only specifies minimum knowledge required from each of the entities in order to model a situation. In contrast, it does not specify the limit of this knowledge. This feature has two benefits (i) it imposes minimum monitoring capabilities of context-aware architectures and (ii) it provides a flexibility to developers to instantiate only a subset of context parameters depending on their problem domain or availability of sensing technologies.

The generic graphical representation of K*i*CM makes it simple to use and abstracts it from any knowledge representation formalism. This gives developers the flexibility to use different knowledge representation formalisms and subsequently different inference mechanisms. As illustrated in section 6.2.2, knowledge about situations is represented as a rule and as a Bayesian network.

In KoDA, inference rules are not limited to a task required to be accomplished by a context-aware system. In KoDA, inference rules represent occurrence of a situation in which the users may be involved in a number of tasks. Hence, KoDA removes the need of specifying inference rules in context-aware systems which subsequently prevents unnecessary repetitions of inference rules.

The process of adding context-aware systems to be supported in KoDA is simplified. As illustrated in section 6.3, to add a context-aware system a developer simply specifies the names of situations in which the systems should be invoked. This process does not require developers of context-aware systems to familiarise with any knowledge representation language.

KoDA is responsible for gathering data from sensors, interpret it and use the resultant context parameters and other directly acquired context parameters to recognise ongoing situation and invoke appropriate context-aware systems. Hence, KoDA removes the reasoning burden from resource-constrained devices. This implies that even the most miniature and resource-constrained device can be made aware of ongoing situation to appropriately respond.

A number of researchers, including Chen (2004), propose to use information about users' scheduled events from sources such as electronic diaries in order to recognise ongoing situations. This approach, however, relies on the users to specify these events. In case a user forgets, which is most likely, a situation cannot be recognised. KoDA removes the need of relying on this information.

7.6.2 Limitations of KoDA

A model is an approximation of the real world and hence K*i*CM does not provide a mirror image of situations. K*i*CM only combines relevant knowledge about the people and their computer-related activities, location, time and devices to approximate situations. This implies that the knowledge about situations that KoDA possess is intensive but incomplete.

In KoDA, knowledge about situations is encoded by a developer and hence the ability of KoDA to recognise ongoing situations is limited to this knowledge. KoDA cannot infer new situations from the knowledge it possess. In future we plan to extend KoDA to utilise information it gather about the environment and its inferences to derive new situations.

In this research we used five sensors and four situations to evaluate KoDA on the accuracy of situation recognition, where three research students were involved. Whilst the results from this research are promising, further investigations of KoDA which use more sensors and situations and involve more users with different roles are required to confirm the performance of KoDA.

Since inference rules are specified based on context parameters, adding or removing a sensor from KoDA requires a modification of these rules. This limitation can be addressed by specifying inference rules with numerous unused context parameters. This solution, however, may result to performance issues and hence care should be taken when adopting it.

7.7 Summary and Conclusions

This chapter has described two approaches used in this research to evaluate the ability of KoDA to recognise ongoing situation. In the first approach, KoDA was evaluated in a real world environment where a prototype was used to recognise situations of the real users. The aim of this evaluation was to assess the ability of KoDA to use information it gathers from the real world environment to accurately recognise ongoing situations. Hence, this evaluation did not take into account uncertainties. In the second approach the impact of including knowledge of certainty levels of sensors in situation recognition was assessed.

The results of these evaluations significantly support our hypothesis. The results show that when KoDA recognises situations based on limited knowledge of the users, time and location, the accuracy of situation recognition is very low. This

accuracy increases significantly when knowledge of the users' computer-related activities is included in situation recognition. The results also show that this accuracy is further increased when knowledge of certainty level of each sensor is included in situation recognition.

This shows that KoDA takes advantage of accessible information about a physical environment to recognise ongoing situations and invoke appropriate context-aware systems. This information is not limited to a specific task and hence KoDA enables context-aware systems to adapt to social dynamics. KoDA also removes the need of relying on information about the users' scheduled events in order to recognise ongoing situations. Consequently, KoDA makes the use of devices intuitive and hence contributing to the vision of UbiComp.

Chapter 8

Conclusions and Future Work

This chapter provides a summary and conclusion of this research. This chapter provides a summary of the contributions from this research and outlines the benefits of the research outcomes. This chapter also outlines limitations and projects the future directions of this research and the research in context-awareness.

This research concludes that the knowledge of the users and their computer-related activities, the users' devices, location and time is crucial for a context-aware architecture to accurately recognise an ongoing situation. This research also concludes that knowledge of certainty level of each sensor is also crucial for for a context-aware architecture to accurately recognise an ongoing situation.

8.1 Thesis Summary

As the number of devices increases, it is imperative to increase seamless interactions between the users and devices, and to make these devices less intrusive. This is because as more devices emerge, it becomes difficult and more time consuming for the users to interact with and effectively use them. Among the solutions to this problem, researchers have been developing computing systems that are responsive to their environments. In Context-Awareness research domain, these systems are called context-aware systems.

The survey of the existing context-aware systems, section 2.4, shows that few of these systems can respond to a set of context parameters that correspond to a specific task. Since these systems are supported by context-aware architectures, in chapter 3 we conducted their analysis. Our analysis concludes that this is because these architectures are not developed to recognise situations. As a result, these architectures are designed with limited representation of the real *world* in which the users and devices interact. Subsequently, these architectures lack reasoning capability to use available information to recognise ongoing situations and hence to enable context-aware systems to respond to situations.

This research proposes a Knowledge-driven Distributed Architecture (KoDA), chapter 5. To design KoDA, we first developed a Knowledge-intensive Context Model (K*i*CM), chapter 4. K*i*CM is a comprehensive model of the *real world*. It provides a systematic way of identifying and representing knowledge of the relevant entities required to sufficiently represent situations in KoDA. To sufficiently and consistently describe the entities specified in K*i*CM, we developed a

synthesised taxonomy, section 2.5.2. This taxonomy provides an extensive list of common context parameters used in context-awareness research community.

To illustrate the application of KoDA and K*i*CM, this research has implemented a prototype, chapter 6. In this prototype we have illustrated how K*i*CM can be used to represent knowledge about situations in KoDA. Since all entities are equally treated in K*i*CM, computer-related activities of nearby users are taken into account when modelling a situation. This makes the resultant model more realistic. We also illustrated how KoDA can monitor a physical environment, interpret the acquired data and use the resultant information to recognise ongoing situation and automatically invoke required context-aware systems.

To assess on how accurate KoDA is on recognising ongoing situations, we conducted performance evaluation in chapter 7. This evaluation shows that the accuracy of situation recognition increases significantly when knowledge of the users and their computer-related activities, the users' devices, location and time is used. This evaluation also shows that this accuracy is further increased when knowledge of certainty level of each sensor is also used. Section 8.2 provides a summary of the contributions of this research.

8.2 Summary of Contributions

This section summarises the contributions of this research as follows;

8.2.1 Synthesised Taxonomy of Context Parameters

The synthesised taxonomy of context parameters, appendix A, is an enhancement of the existing taxonomies of context parameters. This taxonomy provides a systematic way of identifying and representing knowledge about common entities in Context-Awareness. This taxonomy is comprehensive as it is developed without an influence of the development of any context-aware system. The prime feature of this taxonomy is its classification of context parameters based on the entities. This feature provides a focused lens where numerous context parameters about an entity can be identified and represented.

8.2.2 Knowledge-intensive Context Model

The Knowledge-intensive Context Model (K*i*CM), figure 4.1, is a novel model of a situation. It is developed to systematically identify and represent knowledge about entities required to sufficiently represent a situation. K*i*CM improves the existing theory-based models by including knowledge of the users' computer and their computer-related activities to model a situation. This extension is justified in the *in-situ* evaluation, section 7.4, as it shows that the accuracy of situation recognition is significantly increased when knowledge of the users' computers and computer-related activities is included in situation recognition. The benefits of K*i*CM are provided in section 7.6.1.

8.2.3 Knowledge-driven Distributed Architecture

The Knowledge-intensive Distributed Architecture (KoDA), figure 5.1, is a novel context-aware architecture designed to dynamically recognise ongoing situation. KoDA improves the existing context-aware architectures as it can also reason about information it gathers to recognise ongoing situations. KoDA is designed based on K*i*CM and hence with a comprehensive model of the real world in which the users and devices interact. Hence these situations are not confined to a particular task and thus KoDA enables context-aware systems to respond to social dynamics. As shown in the *in-situ* evaluation, section 7.4, KoDA uses information it gathers about the participants and their computer-related activities, location, time and computers to accurately recognise ongoing situations. The benefits of KoDA are provided in section 7.6.1.

8.3 Benefits of the Research Outcomes

1. The taxonomy can be used by developers of context-aware systems to identify context parameters which may be required to sufficiently represent key entities. The developers can also use the taxonomy to identify appropriate technologies which may be required to monitor these entities.

2. K*i*CM can be used by;

    - Researchers seeking to use K*i*CM for further investigations.

    - Developers seeking to use K*i*CM to identify and represent knowledge about situations.

3. KoDA can be used by;
    - Researchers seeking to implement KoDA for further investigations.
    - Developers seeking to implement KoDA for supporting existing or new context-aware systems.
    - Users benefit from KoDA by being able to intuitively use their devices. With KoDA, for instance, a context-aware recommendation application will recommend settings of devices not only when the user is in a specific venue, but based on ongoing situation.
    - The ability of the users to intuitively use their devices may increase the users' efficiency and productivity in an enterprise. KoDA has also a potential to optimise workflow in enterprises by providing real time information about resources, such as employees and equipments.
4. The results of this research can be used by other researchers seeking to design novel context-aware architectures or to improve exiting ones.

8.4 Research Limitations

Throughout the evaluations, only three research students were involved. Hence KoDA was limited to recognise only situations that involved research students. It is worth investigating how KoDA would recognise other situations that involve users with different roles.

The prototype used in the evaluations cannot differentiate, for instance, if the user is at her desk or elsewhere in the room. Hence, on occasions where a par-

ticipant was in a the room but not interacting with her computer, this research assumed that the participant was at her desk.

Throughout the offline evaluation, we used a specific time period to determine whether the participants are using their computers or not. Although this yields reasonable results, an alternative approach is to consider this as uncertainty and thus to apply fuzziness, as described by McKeever (2011, p. 96).

8.5 Future Directions

The emerging context-aware architectures, like KoDA, are capable of storing information about recognised situations and evidences used to infer these situations. This information can be used as another source of knowledge to these architectures. The majority of the existing inference mechanisms in context-awareness, however, are incapable of learning. Much effort is devoted to develop inference mechanisms that are capable of quantifying and preserving uncertainties on situation recognition. Hence, research on learning mechanisms for knowledge-driven context-aware architectures is required. Researcher can adapt lessons learnt from similar work such as from Munguia Tapia (2008), Ravi *et al.* (2005) and Bao & Intille (2004). Such mechanisms, however, should be able to dynamically use information from multiple sources such as social media websites, usage profiles of devices, smart meters, smart energy and smart grid.

The majority of the existing context-aware architectures are designed to acquire knowledge from other sources such as social media, cloud storages and information kiosks. Such knowledge comes in different formats which may differ from

the internal knowledge representation of a context-aware architecture. The question is; How can new inference rules be dynamically added in a context-aware architecture without conflicting the existing inference rules and subsequently affecting its operations? Research is required to investigate how knowledge from different sources can be transformed and added in the existing knowledge of a context-aware architecture without affecting it.

Context-aware architectures are designed to enable context-aware systems to understand their changing environments and the users' computing needs and hence to effectively use them to support the users. The goal is to increase seamless interactions between the users and computing devices, and to make computing devices less intrusive and hence to reduce the users' cognitive overload. Since the cognitive overload is associated with the users' efficiency in their every day activities, it is worth investigating the impact of context-aware architectures on the users' efficiency.

KoDA makes context-aware systems responsive to situations and hence the physical environment and the people in them *smart*. Its ability to intelligently use available information and dynamically recognise ongoing situations enable context-aware systems to constantly be aware of the desirable settings that devices should adhere to and computing needs that the users may require. With the exponential increase of technology, it is worth investigating how KoDA will perform when more users, with different roles, are involved and when more situations and technologies are used.

With the unprecedented increase of handheld and wearable devices that are packed with sensors, technology will enable people to be more conscious about

their health. Using their handheld devices, people will be able to know how many of calories, for instance, they are about to consume from eating their food and how many physical activities they will need to do in order to remain healthier. In addition, patients will be warned when they are about to overdose themselves or will be notified when they are supposed to take their medications. The technology to monitor the users' activities, such as the *Nike Fuel Band*[1] wearable device, is already in place. Products in supermarkets are already attached with RFID tags. In addition, recently handheld devices have been developed with the capability of reading RFID tags, such as NFC[2] enabled Smartphones.

To date, many navigation systems rely on data from GPS. This trend will change in the feature. Navigation systems will be relying on the data from sensors embedded almost everywhere. Physical buildings, roads, signposts, bins and any object you can think of will be attached with sensors. These sensors will contain information about the objects. Sensors attached to a building, for instance, may contain the name of the building, the names of the streets the building is located, and whether the building is a shopping mall or a hospital. Using handheld devices, this information can be used for navigation purposes such as helping blind people walk or identifying tourist cites. A lot of effort has been devoted to localise objects in indoor environments. With the unprecedented increase of sensors, the same lessons can be adapted in localising objects outdoors.

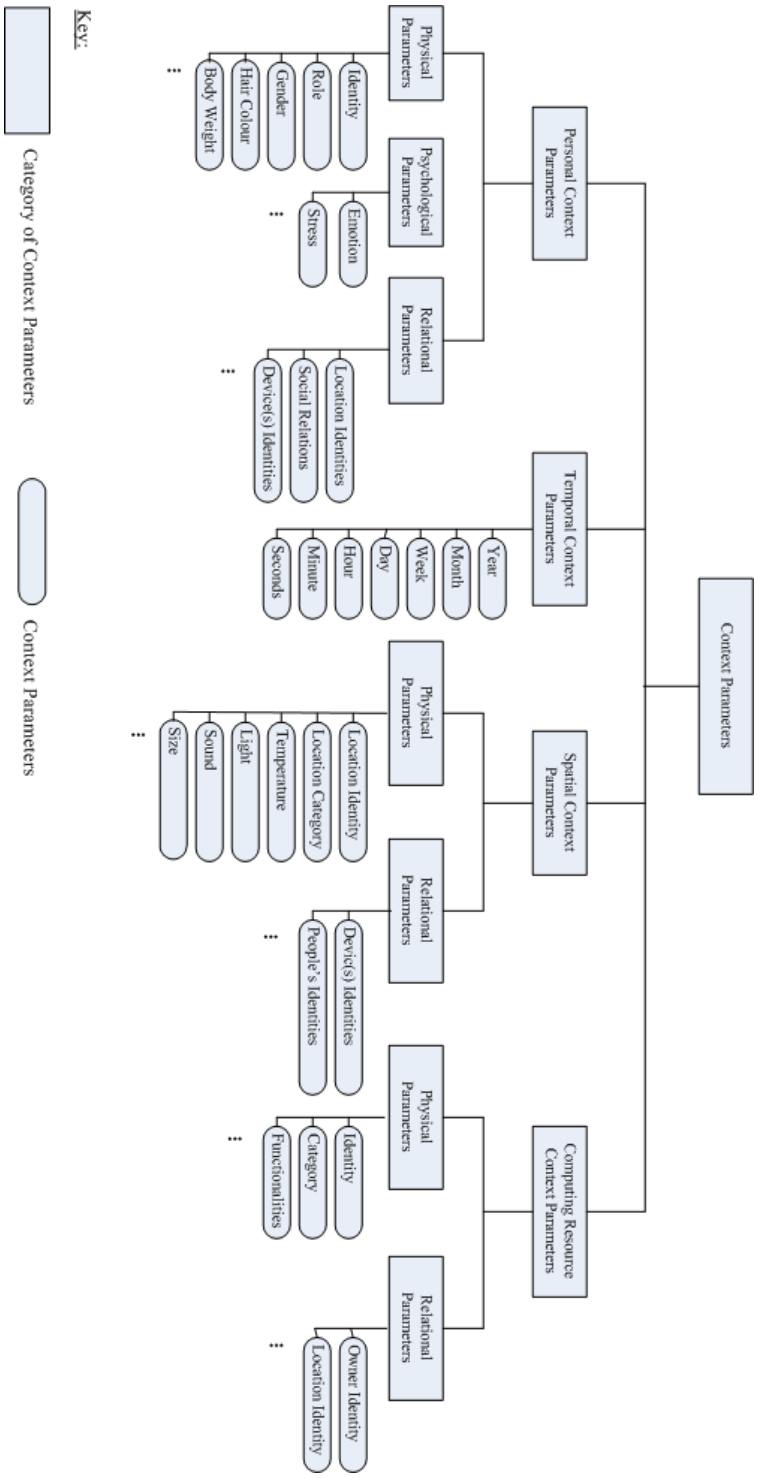
Different sensing technologies are emerging and information sources, such as social media websites, usage profiles of devices, smart meters, smart energy and smart grid, are rapidly increasing. The number of handheld and wearable devices

[1] http://www.nike.com/us/en_us/c/nikeplus-fuelband
[2] NFC is an acronym for Near Field Communication

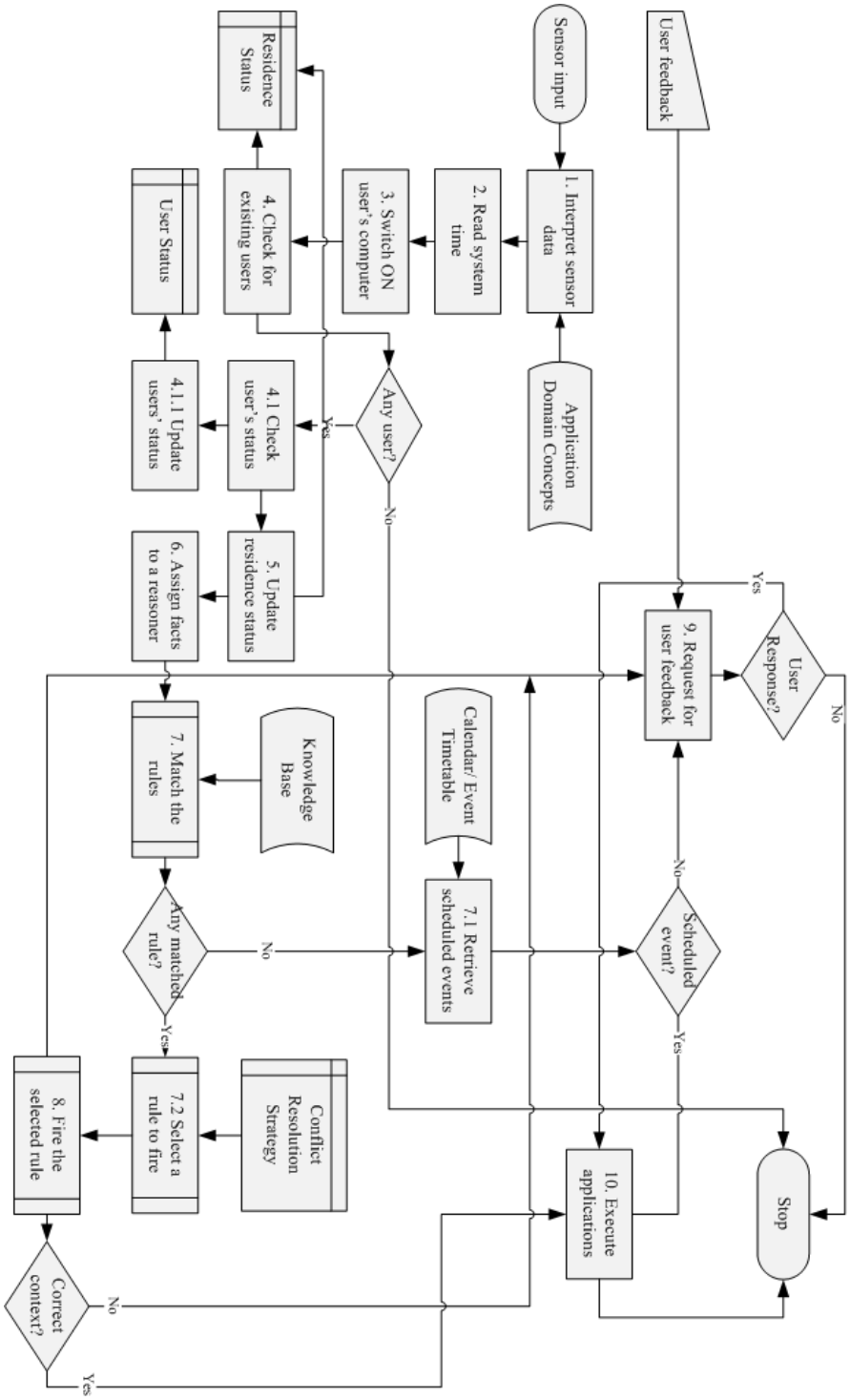
per person is also rapidly increasing. These technologies provide unprecedented access to information about us. Such information may reveal our intentions, medical conditions and our interests. With access to such information, context-aware architectures will make our devices more aware of and responsive to our social life. Devices will become *social-aware*; they will respond as we want and when we want them to. Devices will be able to predict our intentions and determine the best way to support us. In a academic conferences, for instance, the participants' devices will not only seamlessly download the presentation slides of a keynote speaker but will also provide relevant literature based on the participants' tweets. In lecture sessions, students' devices will gather lecture notes and acquire relevant real world examples based on students' cultural, ethnic and geographical background.

Appendices

Appendix A Synthesised-taxonomy of Context Parameters



Appendix B A Process Flow of KoDA



Appendix C Description of the Situations used in this Research

Situation	Description
busy on computer	A situation whereby a research student is at his/her desk in his/her research room while interacting with his/her computer. In this situation, the student may be writing to or reading from the computer.
busy at desk	A situation whereby a research student is at his/her desk in his/her research room without interacting with his/her computer. In this situation, the student may be engaged in activities that do not involve the usage of a keyboard, a mouse or both.
busy working	A situation whereby two or more research students are at their desks in their research room while both of them are interacting with their computers.
working	A situation whereby two or more research students are at their desks in their research room but one or both of them are not interacting with their computers.
informal meeting	A meeting whereby a research student meets with his/her supervisor(s) and/or adviser(s) for informal discussions. This situation occurs when a venue is occupied by other students, who may be interacting with their computers.
semi-formal meeting	A meeting whereby a research student meets with his/her supervisor(s) and adviser(s) for less formal discussions. This situation occurs only when a venue is unoccupied by other students.

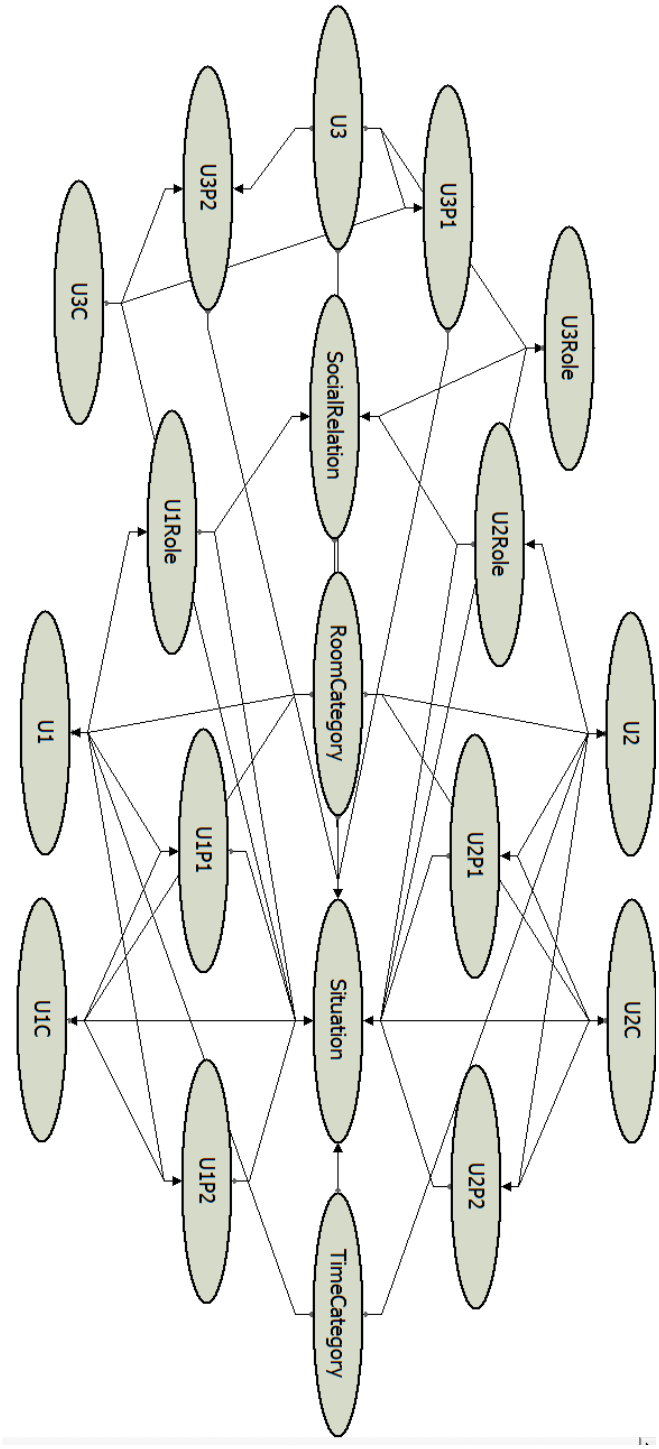
```
rule "2.0 Busy at Desk"
        when
            $roomResidents: List()
            $userStatus: List()
            $userIn: User()
            $deviceList: List()
            $time: Time(timeFor != "AfterWorkingHours")
            Room($venue: roomname, roomcategory == "Research Room")
            $user: ArrayList(size == 1) from collect (User(userrole == "Student", officename == $venue) from $roomResidents)
            not (User(userrole == "Supervisor" || userrole == "Adviser") from $roomResidents)
            Device(username == $user.username, room == $venue, type == "Computer", status in ("ON", "OFF")) from $deviceList

        then
            String situation = drools.getRule().getName().substring(4);
//          AppManager applicationManager = new AppManager(situation, $roomResidents);
            System.out.println("Time: " + $time.time + ", Venue: " + $venue + ", Social Context: " + situation);
    end

rule "2.1 Busy at Desk"
end

rule "3.0 Busy at Desk"
        when
            $roomResidents: List()
            $userStatus: List()
            $userIn: User()
            $deviceList: List()
            $time: Time(timeFor != "AfterWorkingHours")
            Room($venue: roomname, roomcategory == "Research Room")
            $user: ArrayList(size == 1) from collect (User(userrole == "Student", officename == $venue) from $roomResidents)
            not (User(userrole == "Supervisor" || userrole == "Adviser") from $roomResidents)
            Device(username == $user.username, room == $venue, type == "Computer", status == "ON") from $deviceList
            UserStatus(username == $user.username, pname[0].processname == "keyboard", pname[0].status == "false",
            pname[1].processname == "mouse", pname[1].status == null) from $userStatus

        then
            String situation = drools.getRule().getName().substring(4);
//          AppManager applicationManager = new AppManager(situation, $roomResidents);
            System.out.println("Time: " + $time.time + ", Venue: " + $venue + ", Social Context: " + situation);
end
```


U3
U3P2
U3P1
U3C
U3Role
SocialRelation
U1Role
U2Role
RoomCategory
U1
U2
U1P1
U2P1
U1C
Situation
U2C
U1P2
U2P2
TimeCategory

```
public class MouseMonitor {

    private String IP;
    private String hostname;

    public UserProcess checkMouseStatus(String ip) {

        IP = ip;
        hostname = "temp@" + IP;
        UserProcess result = new UserProcess();
        Runtime rt = Runtime.getRuntime();
        result.processname = "mouse";

        try {
            boolean a = ComputerPing.getClientStatus(IP);

            if (a != false) {

                Process proc = rt.exec("../apps/client_mouse_monitor.sh " + hostname);
                InputStreamReader reader = new InputStreamReader(proc.getInputStream());
                BufferedReader bufferedReader = new BufferedReader(reader);
                String line = null;
                String line2 = "";
                while ((line = bufferedReader.readLine()) != null) {
                    line2 += line;
                }

                result.getMap().put("mouse", result.status = (line2 != null && line2.contains("true") ? "true" : "false"));
            }
        } catch (Exception e) {
```

```
public static boolean mouseStatus(Calendar calendar) {
    String lastLine = null;     boolean status = false;     Scanner file_reading = null;
    String before_time = null;  List<String> lines = new ArrayList<String>();
    DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm");
    try{
        for (int x = 0; x < 5; x++){
            calendar.add(Calendar.MINUTE, -x);
            before_time = dateFormat.format(calendar.getTime());
            file_reading = new Scanner(new FileReader("/home/temp/mouse_monitoring/mouse_activity.log"));

            while(file_reading.hasNextLine()){
                lastLine = file_reading.nextLine();
                if (lastLine.contains(before_time)){
                    lines.add(lastLine.substring(27));
                }
            }
            calendar.add(Calendar.MINUTE, x);
        }

        for (int x=0; x < lines.size(); x++){
            int y = x+1;
            if (y == lines.size()){
                break;
            }
            String pCoordinates = lines.get(x);
            String nCoordinates = lines.get(x+1);

            if (!pCoordinates.equalsIgnoreCase(nCoordinates)){
                status = true;
                break;
            }
        }
    }
```

```
public static boolean checkStatus(Calendar calendar) {
    boolean status = false;
    Scanner file_reading = null;
    String before_time = null;
    DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm");
    try {
        for (int x = 0; x < 5; x++){
            calendar.add(Calendar.MINUTE, -x);
            before_time = dateFormat.format(calendar.getTime());
            file_reading = new Scanner(new FileReader("/home/temp/keyboard_monitoring/keyboard_activity.log"));
                while(file_reading.hasNextLine() && !status) {
                    status = file_reading.nextLine().indexOf(before_time) >= 0;
                }
            calendar.add(Calendar.MINUTE, x);
        }
    }
    catch(IOException e) {
        e.printStackTrace();
    }
    finally {
        try { file_reading.close() ; } catch(Exception e) { /* ignore */ }
    }
    return status;
}
```

```
        return userNList;
    }

    public String getUserDetails(String sTag, Element xmlElement) {

        NodeList nList = xmlElement.getElementsByTagName(sTag).item(0).getChildNodes();
        Node nValue = (Node) nList.item(0);
        String nodeValue = nValue.getNodeValue();

        return nodeValue;
    }

    public void readUserDetails(){

        Integer maxNode = userNList.getLength();

        for (int i = 0; i < maxNode; i++) {

            Node nNode = userNList.item(i);
            if (nNode.getNodeType() == Node.ELEMENT_NODE) {

                xmlElement = (Element) nNode;
                String MobileID = this.getUserDetails("TagID", xmlElement);
                String TagID = this.getUserDetails("TagID", xmlElement);
                String UserRole = this.getUserDetails("role", xmlElement);
                String DeviceOwner = this.getUserDetails("name", xmlElement);
                String OfficeName = this.getUserDetails("office", xmlElement);
                String ClientIP = this.getUserDetails("IP", xmlElement);
                String ClientMAC = this.getUserDetails("MAC", xmlElement);
                User user = new User(DeviceOwner, OfficeName, MobileID, UserRole, TagID, ClientIP, ClientMAC);
                this.mapUsers.put(TagID, user);
            }
        }
    }
```

```
public AppManager(String situation, List<User> residents) throws IOException, InterruptedException{

    Runtime rt = Runtime.getRuntime();
    Scanner file_reader = new Scanner(new FileReader("../apps/app_config.txt"));
    String line = file_reader.nextLine();
    while (file_reader.hasNextLine()){
        if (line.contains(situation)){
            String appName = line.substring(0, line.indexOf(situation) - 2);
            for (User user: residents){
                String ip = user.getPcIP();
                try {
                    boolean status = ComputerPing.getClientStatus(ip);
                    if (status){
                        rt.exec("java -cp /home/dennis/Documents/Prototype/apps " + appName + " " + ip);
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
        line = file_reader.nextLine();
    }
    file_reader.close();
}
```

```
public class ServerRun implements ClientListener {

    private final static int SOCKET_PORT = 2000;    private ServerSocket serverSocket;
    StatefulKnowledgeSession ksession = null;       KnowledgeRuntimeLogger logger = null;
    AccessUserInfo userAccessDetails = new AccessUserInfo();       AccessRoomInfo roomAccessDetails = new AccessRoomInfo();
    AccessDeviceInfo deviceAccessDetails = new AccessDeviceInfo();  KeyboardMonitor keyboardMonitor = new KeyboardMonitor();
    MouseMonitor mouseMonitor = new MouseMonitor();

    User user = null;   Room room = null;   Device device = null;   List<Device> userDevices = null;
    String date = null; Time time = new Time(); List<User> residents = null;

    private PrintWriter user_activities_log;    private PrintWriter user_status_updates_log;
    private Map<String, List<User>> mapRoomResidents = new HashMap<String, List<User>>();
    private Map<String, List<UserStatus>> mapRoomUserStatus = new HashMap<String, List<UserStatus>>();
    private Map<String, List<Device>> mapUserDevice = new HashMap<String, List<Device>>();

    public ServerRun() throws Exception {
        this.serverSocket = new ServerSocket(SOCKET_PORT);
        this.user_activities_log = new PrintWriter(new FileWriter("user_activities.log"), true);
        this.user_status_updates_log = new PrintWriter(new FileWriter("user_status_updates.log"), true);
    }

    protected void initiateKBase() throws Exception {
        Inference inference = new Inference();
        KnowledgeBase knowledgeBase = inference.readKnowledgeBase();
        this.ksession = knowledgeBase.newStatefulKnowledgeSession();
    }

    protected void listen() {
        try {
            Socket socketConnection = serverSocket.accept();
            BufferedReader stdIn = new BufferedReader(new InputStreamReader(socketConnection.getInputStream()));
            String proxyInput = stdIn.readLine();

            if (proxyInput != null) {
```

```
public class RfidDetect extends Thread {

    private final static String SERVER_IP = "147.252.229.XXX";
    private final static int SERVER_PORT = 2000;

    private FedmIscReader reader;
    private Long readerID = null;
    private FedmIscReaderInfo readerInfo;

    private int maxNumberTags = 20;
    private boolean stop = false;

    public RfidDetect() throws Exception {
        this.reader = new FedmIscReader();
        this.reader.setTableSize(FedmIscReaderConst.ISO_TABLE, this.maxNumberTags);
        this.reader.connectCOMM(1);
        this.readerInfo = this.reader.getReaderInfo();
        this.readerID = this.readerInfo.getDeviceID();
    }

    public void run() {

        while (!this.stop) {
            try {
                this.reader.disConnect();
            } catch (Exception e) {
                System.err.println("Error disconnecting reader. ERROR: " + e.getMessage());
                e.printStackTrace();
            }
        }
    }

    public static void sendData(String data) throws Exception {
        DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        System.out.println(dateFormat.format(Calendar.getInstance().getTime()) + " Sending: " + data);
        Socket socket = new Socket(SERVER_IP, SERVER_PORT);
        socket.getOutputStream().write(data.getBytes());
        socket.getOutputStream().flush();
        socket.getOutputStream().close();
        socket.close();
        System.out.println(dateFormat.format(Calendar.getInstance().getTime()) + " Sending: " + data + " - SENT");
```

```
public void newTagIdReceived(ReaderEvent event) {
    try {
        String datafromClient = (String) event.getSource();          String tagID = (datafromClient.substring(0, 64));
        String readerID = (datafromClient.substring(64, 65));        user = this.userAccessDetails.getMapUsers().get(tagID);
        room = this.roomAccessDetails.getMapRooms().get(readerID);   device = this.deviceAccessDetails.getMapDevices().get(user.username);

        DateFormat dateformat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");          date = dateformat.format(Calendar.getInstance().getTime());
        Date afterwork = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.ENGLISH).parse(date.substring(0, 10) + " 17:00:00");
        Date workingHours = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.ENGLISH).parse(date.substring(0, 10) + " 08:00:00");
        Date beginlunch = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.ENGLISH).parse(date.substring(0, 10) + " 12:00:00");
        Date endlunch = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.ENGLISH).parse(date.substring(0, 10) + " 14:00:00");
        Date timenow = Calendar.getInstance().getTime();

        if (timenow.after(workingHours)){

            residents = this.mapRoomResidents.get(room.ID);
            if (residents == null) {

                userDevices = this.mapUserDevice.get(room.ID);
                if (userDevices == null) {

                    List<UserStatus> userStatusList = this.mapRoomUserStatus.get(room.ID);
                    if (userStatusList == null) {

                        if (!residents.contains(user)) {
                            System.out.println(date + ": " + user.getUsername() + " was added in the list of the existing users in " + room.name);
                            this.user_activities_log.println(date + ", " + user + " entered in " + room.name);

                            residents.add(user);

                            if (user.userrole.equalsIgnoreCase("Student") && user.officename.equals(room.name)) {
                                if (!ComputerPing.getClientStatus(device.IP)){

                                    ClientON.switchClientON(device.MAC);
                                    device.setStatus("ON");
```

Project Name: A Knowledge-driven Distributed Architecture for Context-Aware Applications

Data about: Ongoing situation and existing users

Date of Data Collection: .....

Name of Data Recorder: .....

Signature of Data Recorder: .....

	$User_1$	$User_2$	$User_3$	$User_4$	$User_5$	Ongoing Situation
09:00						
09:05						
09:10						
09:15						
09:20						
09:25						
09:30						
09:35						
09:40						
09:45						
09:50						
09:55						
10:00						
10:05						
10:10						
10:15						
10:20						
10:25						
10:30						

UC	UP1	UP2	U	WorkingHours	busy on computer	busy at desk	unrecognised
on	true	true	in	yes	0.76	0.24	0.0
				no	0.05	0.05	0.99
			out	yes	0.63	0.07	0.3
				no	0.05	0.05	0.99
		false	in	yes	0.72	0.28	0.0
				no	0.05	0.05	0.99
			out	yes	0.59	0.02	0.39
				no	0.05	0.05	0.99
		null	in	yes	0.61	0.16	0.28
				no	0.05	0.05	0.99
			out	yes	0.56	0.22	0.28
				no	0.05	0.05	0.99
	false	true	in	yes	0.75	0.05	0.2
				no	0.05	0.05	0.99
			out	yes	0.62	0.03	0.35
				no	0.05	0.05	0.99
		false	in	yes	0.01	0.79	0.2
				no	0.05	0.05	0.99
			out	yes	0.01	0.72	0.27
				no	0.05	0.05	0.99
		null	in	yes	0.22	0.63	0.15
				no	0.05	0.05	0.99
			out	yes	0.26	0.61	0.17
				no	0.05	0.05	0.99
	null	true	in	yes	0.67	0.19	0.14
				no	0.05	0.05	0.99
			out	yes	0.64	0.22	0.14
				no	0.05	0.05	0.99
		false	in	yes	0.07	0.64	0.29
				no	0.05	0.05	0.99
			out	yes	0.07	0.61	0.32
				no	0.05	0.05	0.99
		null	in	yes	0.01	0.76	0.23
				no	0.05	0.05	0.99
			out	yes	0.07	0.61	0.32
				no	0.05	0.05	0.99
off	true	true	in	yes	0.0076	0.9924	0.0
				no	0.05	0.05	0.99
			out	yes	0.00737	0.99263	0.0
				no	0.05	0.05	0.99
			.	yes	0.00517	0.99483	0.0

REFERENCES

Abowd, G. (1999). Classroom 2000: An experiment with the instrumentation of a living educational environment. *IBM Systems Journal*, **38**, 508–530.

Abowd, G., Atkeson, C., Hong, J., Long, S., Kooper, R. & Pinkerton, M. (1997). Cyberguide: A mobile contextaware tour guide. *Wireless Networks*, **3**, 421–433.

Adomavicius, G. & Tuzhilin, A. (2008). Context-aware recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, 335–336, ACM.

Anagnostopoulos, C. & Hadjiefthymiades, S. (2010). Advanced fuzzy inference engines in situation aware computing. *Fuzzy sets and systems*, **161**, 498–521.

Baldauf, M., Dustdar, S. & Rosenberg, F. (2007). A survey on context-aware systems. *INTERNATIONAL JOURNAL OF AD HOC AND UBIQUITOUS COMPUTING*, **2**, 263 – 277.

Baltrunas, L., Kaminskas, M., Ludwig, B., Moling, O., Ricci, F., Aydin, A., Lüke, K.H. & Schwaiger, R. (2011). Incarmusic: Context-

aware music recommendations in a car. In *E-Commerce and Web Technologies*, 89–100, Springer.

Bao, L. & Intille, S.S. (2004). Activity recognition from user-annotated acceleration data. In *Pervasive computing*, 1–17, Springer.

Barkhuus, L. & Dey, A. (2003). Is context-aware computing taking control away from the user? three levels of interactivity examined. In *UbiComp 2003: Ubiquitous Computing*, 149–156, Springer.

Biegel, G. (2005). *A Programming Model for Mobile, Context-Aware Applications*. Ph.D. thesis, University of Dublin, Trinity College.

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

Buxton, W. (1997). Living in augmented reality: Ubiquitous media and reactive environments. *Video mediated communication*, 363384.

Callon, M. (1991). Techno-economic networks and irreversibility. *A sociology of monsters: Essays on power, technology and domination*, **38**, 132–161.

Chen, G. & Kotz, D. (2000). A survey of context-aware mobile computing research. Tech. rep., Citeseer.

Chen, H. (2004). *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. Ph.D. thesis, University of Maryland.

Cheverst, K., Davies, N., Mitchell, K., Friday, A. & Efstratiou, C. (2000). Developing a context-aware electronic tourist guide: some issues

and experiences. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 17–24, ACM.

Ciaramella, A., Cimino, M.G., Marcelloni, F. & Straccia, U. (2010). Combining fuzzy logic and semantic web to enable situation-awareness in service recommendation. In *Database and Expert Systems Applications*, 31–45, Springer.

Cimino, M.G., Lazzerini, B., Marcelloni, F. & Ciaramella, A. (2012). An adaptive rule-based approach for managing situation-awareness. *Expert Systems with Applications*, **39**, 10796–10811.

Cooperstock, J.R., Fels, S.S., Buxton, W. & Smith, K.C. (1997). Reactive environments. *Commun. ACM*, **40**, 65–73.

Coutaz, J., Crowley, J.L., Dobson, S. & Garlan, D. (2005). Context is key. *Commun. ACM*, **48**, 49–53.

Csikszentmihalyi, M. & Larson, R. (1992). *The experience of psychopathology: Investigating mental disorders in their natural settings*, chap. Validity and reliability of the experience sampling method, 43–57. Cambridge University Press.

Da, K., Roose, P., Dalmau, M., Nevado, J. & Karchoud, R. (2014). Kali2much: a context middleware for autonomic adaptation-driven platform. In *Proceedings of the 1st ACM Workshop on Middleware for Context-Aware Applications in the IoT*, 25–30, ACM.

Davis, R. & King, J. (1975). An overview of production systems. Tech. rep., Defence Technical Information Center.

de Andrade, M.T.P. (2007). *Architectural support for ubiquitous access to multimedia content*. Ph.D. thesis, University of Porto.

Denning, P. (1980). Acm president's letter: What is experimental computer science? *Communications of the ACM*, **23**, 543–544.

Devlic, A., Reichle, R., Wagner, M., Pinheiro, M.K., Vanrompay, Y., Berbers, Y. & Valla, M. (2009). Context inference of users' social relationships and distributed policy management. In *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, 1–8, IEEE.

Dey, A. & Abowd, G. (2000a). Towards a better understanding of context and context-awareness. In *CHI 2000 workshop on the what, who, where, when, and how of context-awareness*, 304–307.

Dey, A., Salber, D., Abowd, G. & Futakawa, M. (1999). The conference assistant: Combining context-awareness with wearable computing. In *Proceedings of the 3rd International Symposium on Wearable Computers*, 21–28.

Dey, A.K. (2000). *Providing Architectural Support for Building Context-Aware Applications*. Ph.D. thesis, Georgia Institute of Technology.

Dey, A.K. & Abowd, G.D. (2000b). Cybreminder: A context-aware system for supporting reminders. In *Handheld and Ubiquitous Computing*, 172–186, Springer.

Dix, A., Rodden, T., Davies, N., Trevor, J., Friday, A. & Palfreyman, K. (2000). Exploiting space and location as a design framework for interactive mobile systems. *ACM Trans. Comput.-Hum. Interact.*, **7**, 285–321.

Dockhorn Costa, P., Almeida, J., Ferreira Pires, L. & Van Sinderen, M. (2007). Situation specification and realization in rule-based context-aware applications. In *Distributed Applications and Interoperable Systems*, 32–47, Springer.

Don, B. (1994). The leaps algorithms. Tech. rep., Department of Computer Science, The University of Texas.

Dourish, P. (2004). What we talk about when we talk about context. *Personal and ubiquitous computing*, **8**, 19–30.

Ericsson (2011). More than 50 billion connected devices. Tech. rep., Ericsson.

Evans, D. (2011). The internet of things how the next evolution of the internet is changing everything. Tech. rep., Cisco.

Forgy, C. (1979). *On the Efficient Implementation of Production Systems*. Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University.

Forgy, C. (1982). Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, **19**, 17–37.

Froehlich, J., Chen, M.Y., Consolvo, S., Harrison, B. & Landay, J.A. (2007). Myexperience: a system for in situ tracing and capturing of user feedback on mobile phones. In *Proceedings of the 5th international conference on Mobile systems, applications and services*, 57–70, ACM.

Gellersen, H.W., Schmidt, A. & Beigl, M. (2002). Multi-sensor context-awareness in mobile devices and smart artifacts. *Mobile Networks and Applications*, **7**, 341–351.

Göker, A. & Myrhaug, H. (2002). User context and personalisation. In *Workshop proceedings for the 6th European Conference on Case Based Reasoning*.

Gomes, D., Gonçalves, J.M., Santos, R.O. & Aguiar, R. (2010). Xmpp based context management architecture. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, 1372–1377, IEEE.

Gregory, F. (1993). Cause, effect, efficiency and soft systems models. *Journal of the Operational Research Society*, 333–344.

Gu, T., Pung, H.K. & Zhang, D.Q. (2004a). A bayesian approach for dealing with uncertain contexts. In *Second International Conference on Pervasive Computing*.

Gu, T., Pung, H.K. & Zhang, D.Q. (2004b). Toward an osgi-based infrastructure for context-aware applications. *Pervasive Computing, IEEE*, **3**, 66–74.

Gu, T., Pung, H.K. & Zhang, D.Q. (2005). A service-oriented middleware for building context-aware services. *Journal of Network and computer applications*, **28**, 1–18.

Haghighi, P.D., Krishnaswamy, S., Zaslavsky, A. & Gaber, M.M. (2008). Reasoning about context in uncertain pervasive computing environments. In *Smart Sensing and Context*, 112–125, Springer.

Han, L., Jyri, S., Ma, J. & Yu, K. (2008). Research on context-aware mobile computing. In *Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008. 22nd International Conference on*, 24–30.

Harter, A., Hopper, A., Steggles, P., Ward, A. & Webster, P. (2002). The anatomy of a context-aware application. *Wireless Networks*, **8**, 187–197.

Heckerman, D. (1998). *A tutorial on learning with Bayesian networks*. Springer.

Henricksen, K. (2003). *A Framework for Context-Aware Pervasive Computing Applications*. Ph.D. thesis, School of Information Technology and Electrical Engineering, The University of Queensland.

Hong, J.y., Suh, E.h. & Kim, S.J. (2009). Context-aware systems: A literature review and classification. *Expert Systems with Applications*, **36**, 8509–8522.

Igira, F. & Gregory, J. (2009). Cultural historical activity theory. *Handbook of research on contemporary theoretical models in information systems*, 434–454.

Intille, S.S., Rondoni, J., Kukla, C., Ancona, I. & Bao, L. (2003). A context-aware experience sampling tool. In *CHI'03 extended abstracts on Human factors in computing systems*, 972–973, ACM.

Jensen, F.V. (1996). *An introduction to Bayesian networks*, vol. 210. UCL press London.

Kaenampornpan, M. (2009). *A Context Model, Design Tool and Architecture for Context-Aware Systems Design*. Ph.D. thesis, Department of Computer Science, University of Bath.

Kaptelinin, V. & Nardi, B.A. (1997). Activity theory: basic concepts and applications. In *CHI '97 extended abstracts on Human factors in computing systems: looking to the future*, CHI EA '97, 158–159, ACM, New York, NY, USA.

Kofod-Petersen, A. (2007). *A Case-Based Approach to Realising Ambient Intelligence among Agents*. Ph.D. thesis, Department of Computer and Information Science, Norwegian University of Science and Technology.

Korb, K.B. & Nicholson, A.E. (2003). *Bayesian artificial intelligence*. cRc Press.

Kortuem, G., Kawsar, F., Fitton, D. & Sundramoorthy, V. (2010). Smart objects as building blocks for the internet of things. *Internet Computing, IEEE*, **14**, 44–51.

Kukkonen, J., Lagerspetz, E., Nurmi, P. & Andersson, M. (2009). Betelgeuse: A platform for gathering and processing situational data. *Pervasive Computing, IEEE*, **8**, 49–56.

Larson, R. & Csikszentmihalyi, M. (1983). The experience sampling method. *New Directions for Methodology of Social & Behavioral Science*.

Lee, S., Kim, J., Wang, H., Bae, D., Lee, K., Lee, J. & Jeon, J. (2006). Architecture of rete network hardware accelerator for real-time context-aware system. In *Knowledge-Based Intelligent Information and Engineering Systems*, 401–408, Springer.

Lee, Y.S. & Cho, S.B. (2013). A mobile picture tagging system using tree-structured layered bayesian networks. *Mob. Inf. Syst.*, **9**, 209–224.

Li, Z.Y., Park, J.C., Lee, B. & Youn, H.Y. (2013). Situation awareness based on dempster-shafer theory and semantic similarity. In *Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference on*, 545–552, IEEE.

Lindley, D.V. & Phillips, L. (1976). Inference for a bernoulli process (a bayesian view). *The American Statistician*, **30**, 112–119.

Liu, H. (2010). Biosignal controlled recommendation in entertainment systems. *Technische Universiteit Eindhoven, Eindhoven*, 1–133.

Logan, B., Healey, J., Philipose, M., Tapia, E.M. & Intille, S. (2007). *A long-term evaluation of sensing modalities for activity recognition*. Springer.

Ludford, P.J., Frankowski, D., Reily, K., Wilms, K. & Terveen, L. (2006). Because i carry my cell phone anyway: functional location-based reminder applications. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, 889–898, ACM.

Lueg, C. (2002). On the gap between vision and feasibility. In *Pervasive Computing*, 45–57, Springer.

Lyu, C.H., Choi, M.S., Li, Z.Y. & Youn, H.Y. (2010). Reasoning with imprecise context using improved dempster-shafer theory. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, vol. 2, 475–478, IEEE.

Marmasse, N. & Schmandt, C. (2000). Location-aware information delivery with commotion. In *Handheld and Ubiquitous Computing*, 157–171, Springer.

McKeever, S. (2011). *Recognising Situations Using Extended Dempster-Shafer Theory*. Ph.D. thesis, School of Computer Science and Informatics, National University of Ireland.

Miluzzo, E. (2011). *Smartphone Sensing*. Ph.D. thesis, Dartmouth College.

Miranker, D. (1987). Treat: a better match algorithm for ai production systems. In *Proceedings of the sixth National conference on Artificial intelligence-Volume 1*, 42–47, AAAI Press.

Mitchell, K. (2002). *Supporting The Development of Mobile Context-Aware Systems*. Ph.D. thesis, Computing Department, Lancaster University.

Munguia Tapia, E. (2008). *Using machine learning for real-time activity recognition and estimation of energy expenditure*. Ph.D. thesis, Massachusetts Institute of Technology.

Newell, A. (1973). Production systems: Models of control structures. Tech. rep., Defence Technical Information Center.

Newell, A. (1982). The knowledge level. *Artificial Intelligence*, **1**, 87–127.

ODriscoll, C., MacCormac, D., Deegan, M., Mtenzi, F. & OShea, B. (2008). Rfid: An ideal technology for ubiquitous computing? *Ubiquitous Intelligence and Computing*, 490–504.

Paspallis, N., Rouvoy, R., Barone, P., Papadopoulos, G.A., Eliassen, F. & Mamelli, A. (2008). A pluggable and reconfigurable architecture for a context-aware enabling middleware system. In *On the Move to Meaningful Internet Systems: OTM 2008*, 553–570, Springer.

Raento, M., Oulasvirta, A., Petit, R. & Toivonen, H. (2005). Contextphone: A prototyping platform for context-aware mobile applications. *Pervasive Computing, IEEE*, **4**, 51–59.

Ranganathan, A. & Campbell, R.H. (2003a). An infrastructure for context-awareness based on first order logic. *Personal and Ubiquitous Computing*, **7**, 353–364.

Ranganathan, A. & Campbell, R.H. (2003b). A middleware for context-aware agents in ubiquitous computing environments. In *Middleware 2003*, 143–161, Springer.

Ranganathan, A., Al-Muhtadi, J. & Campbell, R.H. (2004). Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing*, **3**, 62–70.

Ravi, N., Dandekar, N., Mysore, P. & Littman, M.L. (2005). Activity recognition from accelerometer data. In *AAAI*, vol. 5, 1541–1546.

Roalter, L., Kranz, M. & Möller, A. (2010). A middleware for intelligent environments and the internet of things. In *Ubiquitous Intelligence and Computing*, 267–281, Springer.

Russell, D.M., Streitz, N.A. & Winograd, T. (2005). Building disappearing computers. *Commun. ACM*, **48**, 42–48.

Santos, E., Gu, Q. & Santos, E.E. (2011). Incomplete information and bayesian knowledge-bases. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, 2989–2995, IEEE.

Schilit, B. (1995). *A System Architecture for Context-Aware Mobile Computing*. Ph.D. thesis, Graduate School of Arts and Sciences, Columbia University.

Schilit, B. & Theimer, M. (1994). Disseminating active map information to mobile hosts. *Network, IEEE*, **8**, 22 –32.

Schilit, B., Adams, N. & Want, R. (1994). Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, 85 –90.

Schilit, B.N., LaMarca, A., Borriello, G., Griswold, W.G., McDonald, D., Lazowska, E., Balachandran, A., Hong, J. & Iver-

son, V. (2003). Challenge: Ubiquitous location-aware computing and the place lab initiative. In *Proceedings of the 1st ACM international workshop on Wireless mobile applications and services on WLAN hotspots*, 29–35, ACM.

Schmidt, A. & Van Laerhoven, K. (2001). How to build smart appliances? *Personal Communications, IEEE*, **8**, 66–71.

Schmidt, A., Aidoo, K., Takaluoma, A., Tuomela, U., Van Laerhoven, K. & Van de Velde, W. (1999a). Advanced interaction in context. In *Handheld and ubiquitous computing*, 89–101, Springer.

Schmidt, A., Beigl, M. & Gellersen, H.W. (1999b). There is more to context than location. *Computers & Graphics*, **23**, 893–901.

Shafer, S., Krumm, J., Brumitt, B., Meyers, B., Czerwinski, M. & Robbins, D. (1998). The new easyliving project at microsoft research. In *Proceedings of the 1998 DARPA/NIST Smart Spaces Workshop*, 127–130.

Siewiorek, D., Smailagic, A., Furukawa, J., Krause, A., Moraveji, N., Reiger, K., Shaffer, J. & Wong, F.L. (2003). Sensay: A context-aware mobile phone. *ISWC\'03*.

Sohn, T., Li, K.A., Lee, G., Smith, I., Scott, J. & Griswold, W.G. (2005). *Place-its: A study of location-based reminders on mobile phones*. Springer.

Sowa, J. (1991). Principles of semantic networks.

Soylu, A., Causmaecker, P.D. & Desmet, P. (2009). Context and adaptivity in pervasive computing environments: Links with software engineering and ontological engineering. *Journal of Software*, **4**, 992–1013.

Strang, T. & Linnhoff-Popien, C. (2004). 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*.

Streitz, N., Rocker, C., Prante, T., van Alphen, D., Stenzel, R. & Magerkurth, C. (2005). Designing smart artifacts for smart environments. *Computer*, **38**, 41 – 49.

Strobbe, M., Van Laere, O., Dhoedt, B., De Turck, F. & Demeester, P. (2012). Hybrid reasoning technique for improving context-aware applications. *Knowledge and Information systems*, **31**, 581–616.

Studer, R., Benjamins, V.R. & Fensel, D. (1998). Knowledge engineering: Principles and methods.

Tichy, W. (1998). Should computer scientists experiment more? *Computer*, **31**, 32–40.

Truong, B.A., Lee, Y.K. & Lee, S.Y. (2005). Modeling and reasoning about uncertainty in context-aware systems. In *e-Business Engineering, 2005. ICEBE 2005. IEEE International Conference on*, 102–109.

van de Westelaken, R., Hu, J., Liu, H. & Rauterberg, M. (2011). Embedding gesture recognition into airplane seats for in-flight entertainment. *Journal of Ambient Intelligence and Humanized Computing*, **2**, 103–112.

Van Kasteren, T., Noulas, A., Englebienne, G. & Kröse, B. (2008). Accurate activity recognition in a home setting. In *Proceedings of the 10th international conference on Ubiquitous computing*, 1–9, ACM.

Wang, X.H., Zhang, D.Q., Gu, T. & Pung, H.K. (2004). Ontology based context modeling and reasoning using owl. 18–22.

Want, R. (2004). Enabling ubiquitous sensing with rfid. *Computer*, **37**, 84–86.

Want, R., Hopper, A., Falcão, V. & Gibbons, J. (1992). The active badge location system. *ACM Trans. Inf. Syst.*, **10**, 91–102.

Weiser, M. (1991). The computer for the 21st century. *Scientific American*.

Weiser, M. (1994). The world is not a desktop. *interactions*, **1**, 7–8.

Woods, W. (1975). What's in a link: Foundations for semantic networks. Tech. rep., DTIC Document.

Yau, S.S. & Karim, F. (2004). An adaptive middleware for context-sensitive communications for real-time applications in ubiquitous computing environments. *Real-Time Systems*, **26**, 29–61.

Ye, J., Coyle, L., Dobson, S. & Nixon, P. (2007). Using situation lattices to model and reason about context. In *Modeling and Reasoning in Context (MRC) with Special Session on the Role of Contextualization in Human Tasks (CHUT) which is held in conjunction with CONTEXT*, 1–12.

Zhang, D., Guo, M., Zhou, J., Kang, D. & Cao, J. (2010). Context reasoning using extended evidence theory in pervasive computing environments. *Future Generation Computer Systems*, **26**, 207–216.

Zimmermann, A., Lorenz, A. & Oppermann, R. (2007). An operational definition of context. In *Modeling and using context*, 558–571, Springer.