UNIVERSITY
of
GLASGOW

Department of
Computing Science

The Development and Evaluation of a Prototyping Environment for Context-Sensitive Mobile Computing Interaction

Yun-Maw Kevin Cheng

Submitted for the degree of Doctor of Philosophy

February 2003

ABSTRACT

Recent developments in wireless communication, mobile computing, and sensor technologies have prompted a new vision of the world in which we live. As witnesses the effects of Moore's law, which are evident in many aspects of innovative technical opportunity, such as cost, size, capacity, bandwidth, etc. These advances allow us to build new types of human-computer-environment interaction in augmented physical spaces. Ideally, mobile computing devices can go with people so that they can access information on the move as being constantly connected to the digital space. Sensor technologies enable mobile computing devices to sense their users and environments. This increases the interaction bandwidth between a human and a mobile computing device.

Many researchers have made claims about the potential benefits of context-sensitive mobile computing since its first appearance about a decade ago. Quite a few context-sensitive mobile computing applications have been built to demonstrate the usefulness of this technology. With context-sensitive capability, the computation can be responsive according to the environment. The initial focus in context-sensitive mobile computing was on "follow-me" applications in which the presentation and dissemination of information was tailored to the user's tasks on the move. More recently, the focus has moved from systems that sense the location of users towards architectures that help the users sense systems and other physical objects in their immediate environment. In the meanwhile, the proliferation of wireless-communication-enabled mobile computing devices has triggered a number of researchers in this area to consider the importance of heterogeneous development approach between various hardware and software platforms of mobile computing devices and remote information servers. The emergence and impact of this brand-new type of interaction not only depends on a revolution in the underlying technology but also requires fundamental changes in the way that people access information in their daily life.

The development of context-sensitive mobile computing systems requires considerable engineering skills. None of the existing approaches provides an effective means of obtaining location and environmental information using "standard" hardware and software. This raises the entry level of discovering more about this type of interaction to the designers. In addition, it is important to stress that relatively little is known about the usability problems that might arise from interaction with these different context-sensitive mobile computing applications. The focus of this thesis is on the development of a prototyping environment for context-sensitive mobile computing. This thesis makes two contributions. The most significant contribution is the presentation of the Glasgow Context Server (GCS). It has been specifically designed to address the concerns mentioned above. It successfully integrates an off-the-shelf radio Local Area Network (LAN) with the infrared sensors that have been a feature of many previous context-sensitive mobile computing applications. The GCS is intended to help interface designers validate the claimed benefits of location sensing, location disclosing and environment sensing applications. The second contribution is the working applications, in particular, a web-based annotation system for physical objects and a shopping assistant built upon the GCS environment. These demonstrations are used to evaluate the GCS approach and point out the challenging issues in computing technology as well as usability concern. The hope is that this research can provide interface designers with an in-depth reference to a prototyping environment for context-sensitive mobile computing applications.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank my supervisor Professor Chris Johnson for his guidance, enthusiasm and support during this research. I would like to thank David Morning for helping me regarding infrared and radio frequency transmitter development issues. Also, I would like to thank B&Q Braehead, Glasgow for cooperating on the MapView Shopping Assistant case study.

Finally, I would like to thank my family for their support throughout my work on this thesis. Especial thanks goes to Sandra Lu, who provides me encouragement and inspiration during the time I spent on this thesis. Thanks to Dr. Minli Yang for her lovely multi-culture cuisine. It is always a cheer when I run out of steam.

This research was supported by the CSMS Faculty Postgraduate Research Scholarships.

DECLARATION

I hereby declare that this thesis has been composed by myself, that the work herein is my own except where otherwise stated, and that the work presented has not been presented for any other university degree before.

The design of the Glasgow Context Server (GCS) in Chapter 4 and further development in Chapter 9 contain a revised version of material published in the book, Human Factors and Web Development (Johnson and Cheng, 2002). Chapter 6 contains material published in IHM-HCI 2001 (Cheng and Johnson, 2001). These were jointly authored with Chris Johnson. This thesis only exploits those parts of these publications that are directly attributed to the author.

Yun-Maw Kevin Cheng

CHAPTER 1
INTRODUCTION

This chapter provides a general introduction to the aims of the thesis. Firstly, the trend of adopting context information in a new form of human computer interaction is described and discussed in an attempt to express the background to this research. Secondly, the concepts in this type of interaction are discussed. We are not dealing with concept from user modelling and automatic recognizing user's tasks. These are research areas in their own right (Byun and Cheverst, 2001, Jameson, 2001). Thirdly, we introduce taxonomy of this interaction. It is discussed in order to generalise the current researches in this field. Finally, the research aims and plans are presented.

1.1 TRENDS SHAPING HUMAN COMPUTER INTERACTION

Human computer interaction (HCI) is changing. Increasing numbers of users are learning to exploit mobile computing devices such as Personal Digital Assistants (PDAs) in addition to conventional desktop computers. The trend is that users are moving away from tradition desktop computing environments toward mobile and ubiquitous computing environments (Dryer et al., 1999). Mobile computing devices such as mobile phones and wireless enabled PDAs help people to traverse between physical and digital space. Mobile computing devices, wireless and location-sensing technologies create a link between users, physical places, and digital services. These devices enable users to access information from remote servers about the physical location that they are currently in. Such developments form part of a more general trend that is pushing physical and digital environments closer and closer together (Caswell and Debaty, 2000, Hohl et al., 1999, Oppermann and Specht, 2000, Spohrer, 1999). A growing realization is that more and more digital services will play a greater part in our everyday life (Schmandt et al., 2000). For instance, chat rooms and web-based shopping malls duplicate the physical experience within a digital format. It is a matter of personal experience as to whether these digital developments enhance or parody their physical counterparts (Cheng and Johnson, 2001). Many of these developments have, however, been ad hoc. In consequence, a number of HCI researchers have searched for a more systematic approach to the integration of physical and digital space (Ishii and Ullmer, 1997, network, 2002, Want and Russell, 2000). This work has developed sophisticated theories and a number of case studies that point to the benefits of tangible bits and augmented reality. A smaller group of studies have pointed to the ethical problems and social exclusion that might result from such developments (Dryer et al., 1999).

The underlying concept envisioned for mobile and ubiquitous computing researchers is that computers and digital services are widespread and embedded within physical space. Ideally, users in this type of environment can access whatever information and services they want anytime and anywhere. It is obvious that the systems in this type of environment need to be context-sensitive because of the

rapidly changing situation. Users and computing devices are nomadic. The user's needs and computation states are changing over time. Computing services in this type of environment should provide information regarding the situation of their users and environmental conditions. Conventional HCI seems too limited to suit this non-traditional, off-the-desktop computing environment (Dey, 2000, Korkea-aho, 2000, Lieberman and Selker, 2000). It takes input that is explicitly supplied by the users and performs responsive actions upon that input to produce an explicit output. Researchers in the field of HCI have been interested in location-sensing as means of supporting context sensitivity especially in mobile computing, for a considerable period of time. Location information is utilized as an index to tailor the information for the users on the move. The Active Badge system was conceived in the early 1990s and used a network of infrared "base sensors" to locate the badges that users wore as they moved around within a building (Want et al., 1992). The subsequent ParcTab system extended this initial idea. The ParcTab devices that resemble pagers replaced the badges. These provided location information to the system and also offered their users a limited range of services as they moved within a building (Schilit, 1995). More recently, the focus has moved from systems that sense the location of their users towards architectures that help users sense systems and other objects in their environment. This work has been carried on in an attempt to integrate physical and digital space (Kindberg et al., 2000, Leonhardi and Bauer, 2000, Want et al., 1999, Zimmerman, 1999). In consequence, the need to build context-sensitive mobile computing system architectures has emerged. The aim is that it can provide more useful computational services and reduce the amount of explicit input provided by the users (Anhalt et al., 2001, Dey, 2001, Schmidt, 2000, Selker and Burleson, 2000, Tuulari, 2000).

1.2 TECHNICAL FOUNDATION

In section 1.1, we described the actors in this new form of interaction; users, mobile computing devices, wireless communications, sensors, and surrounding objects such as computers, printers, etc. As an idealised scenario, the user might use a wireless network to access digital information related to their immediate environment through a mobile computing device, which can recognize her current surroundings using sensors. From our point of view, there are a number of different interactions involved in this scenario. The mobile computing device displays information about the user's current location. This is similar to scenarios in Augmented Reality (Schmidt and Beigl, 1998). Apart from the traditional graphical user interface (GUI) served between the user and the mobile computing device, the interaction between the surrounding computing resources is similar to Ubiquitous Computing (Weiser, 1996). The computation is based on the user's activities. The implications of this are that the computer system can understand the meaning of user's physical movements and take them as input to the system. This resembles Tangible Bits (Ishii and Ullmer, 1997). The mobile computing device has to be context-sensitive in terms of recognizing their user's natural skills and abilities in order to response appropriately. The concept of this is well explained in the Context-Aware Computing (Schmidt, 2000).

The following section presents research work that forms the base of the context-sensitive mobile computing. A number of the terms used above are described in more detail.

1.2.1 MOBILE COMPUTING

In the traditional wired and stationary paradigm, users have to unplug their computer from one local area network, bring it with them, and plug it back to another local area network when they need to move from one place to another. In contrast, mobile computing technology enables users to continue accessing information while they on the move. A mobile computing environment typically depends upon based stations that are connected to conventional wired networks, or on mobile stations that connect to the network via wireless communication through the base stations. Each station forms a communication cell within which the users' mobile computing device can connect to the network. A number of researchers in this area address the question of "How to deliver relevant information effectively to the mobile devices?" It generates the foundation of much current research in context-sensitive mobile computing (Chen and Kotz, 2000, E. and Bhargava, 1994, Forman and Zahorjan, 1994).

1.2.2 AUGMENTED REALITY

Augmented Reality (AR) was proposed in contrast to Virtual Reality (VR). VR is the idea that users interact with a synthetic imaginary space. Instead, AR enables users to perceive the physical environment as well as virtual entities generated by computer systems superimposed upon or blending with it (Ishii and Ullmer, 1997). It is similar to video special effects, but it is also marked by important differences. AR supports the perceptions of real special effects (Spohrer, 1999). It means special effects turn up right away where the user is currently in a particular place. The visualisation of digital information overlapping upon physical entities is powered by VR devices, such as head-mounted display and goggles. Until recently, the development in mobile computing devices, such as palm or hand-held PC, leads the other AR approach (Cambridge, 2001, Greenhalgh et al., 2001). Figure 1.1 illustrates an AR application, which assists its user to operate the digital camera, runs on a palm-size mobile computing device.

Figure 1.1 Augmented Reality Mobile Computing Application

(Source: http://www.ar-pda.de/index_e.htm)

AR can help to extend human perception by supplying useful and relevant information superimposed upon the physical space where they are located, unlike devices, such as a binocular, that can only extend human perception of seeing what actually exists. An AR system can present the user with relevant information about the user's current situation in a particular location to support user tasks. As an example, an AR tour guide application can present information about the place or the artefact that the user is looking at. In addition to the mobile computing approach, an AR device should be able to sense the user's vicinity using sensors such as Global Positioning System (GPS).

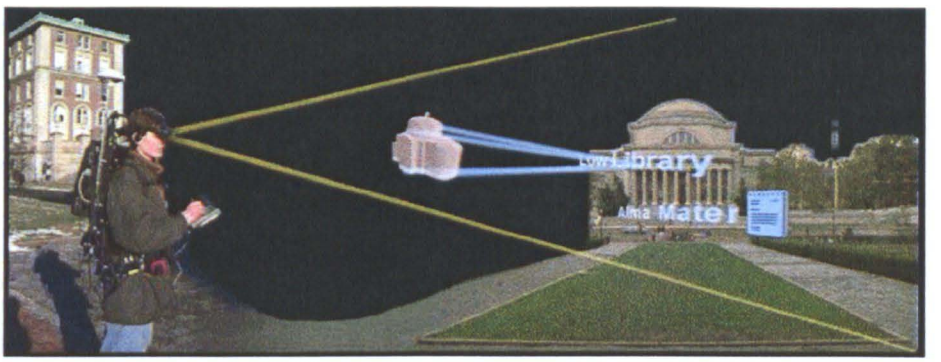


Figure 1.2 Augmented Reality Tour Guide

(Source: http://www.cs.columbia.edu/graphics/projects/mars/mars.html)

1.2.3 UBIQUITOUS COMPUTING

Weiser envisioned that there will be many surroundings, ranging from household appliances to pieces of furniture, equipped or embedded with computers in our daily life. According to Weiser, a new term was created and called Ubiquitous Computing (Ubicomp). It aims to enrich computer usage by making available computers throughout the physical environment (Weiser, 1993). Ubicomp attempts to revolutionize everyday life by allowing computers to vanish into the background while users are using them (Weiser, 1996). This research involves not only hardware components and network connections in mobile computing but also application design for this type of computing environment.

Ubicomp is different from the VR approach, which confines users to the computer synthetic digital environment. In the ubicomp approach, a user is still engaged in their surroundings while being attracted by a particular source of information. The ParcTab, Pad, and Board were the prototypes to embody the vision of ubicomp (Want et al., 1995, Weiser, 1996). ParcTab, as mentioned, is a pager-like device. A Pad is a note-book-sized device. These devices are carried by the users for sharing information wherever they go. A Board is installed in physical place such as a room to display information for group discussion (Michahelles, 2000, Weiser, 1993). The idea was to equip as many people and place with these devices in order to achieve computing everywhere. This would provide users with an illusion that using the devices to do tasks such as reading or writing emails, documents sharing with other colleagues, etc. without knowing the existence of computation.

The function of ParcTab and the Pad devices can be categorised in the mobile computing paradigm. However, in ubicomp applications, these devices not only provide their users with tailored information based on their current location as an AR system can, but are also capable of sensing surrounding computing resources for their users when needed. In order to make the sensing smoother, Weiser also proposed Calm Technology (Weiser, 1998, Weiser and Brown, 1996). This aims to reduce information overload by letting users choose what information is in their attention and what is peripheral. Weiser et al used the Iceberg Model to explain their idea (Weiser, 1998). The part of iceberg above the surface is in the user's attention and the part below the surface is in the periphery. This relaxes the user by moving unneeded information to the border of an interface and also allows more information be ready for using when needed.

1.2.4 TANGIBLE BITS

Ishii gave an interesting example that inspired him to make an effort to link physical and digital space together. In ancient China (even now), people use the abacus to deal with calculations in their daily life. Using an abacus, the user has to physically stir the beads for calculating. The calculation result is formed by the beads in a particular pattern on the abacus. These insights inspired the recent proposed research topic, Tangible Bits (Ishii and Ullmer, 1997). According to Ishii, we operate in two different worlds, physical (atoms) and digital (bits). He argued that the existing interactive systems exploit very little human natural skills and abilities. Instead, most of the current interactive channels between a human and a computer system focus on mice and keyboards. He proposed to treat physical objects as inputs, interfaces, and physical representation of digital information rather than focusing on artificial visual augmentations. The objective of this research is to provide mediums, which are close to human natural skills and abilities, for people to operate computing systems (Ishii and Ullmer, 1997).

Ubicomp focuses on the relationship between user's activities and the immediate environment. However, Tangible Bits is more about the physical manipulations needed to control computer systems (Dourish, 2001). The consensus of these two research trains is that computer systems should be able to understand the meaning of user's physical activities in certain situations. Computer systems can respond more appropriately so their users can concentrate on their tasks and are therefore less aware of the existence of computer systems at hand or nearby. This research, however, raises issues in hardware design in terms of mapping digital information to relevant physical entities. Researchers in this area have to design physical "things" that provide the interface for users to access information.

1.2.5 CONTEXT-AWARE COMPUTING

As mentioned previously, in the mobile or ubiquitous computing environment, the user's location and environmental factors, such as surrounding objects and the computation services, change over time. This poses a challenge to the interaction between the computer systems and their users. Users of this

type of computing systems have to pay more attention to changes of the physical environment and objects within than in the stationary computer systems scenario in order to use the systems properly (Anhalt et al., 2001). Under this circumstance, the context information about the user's current physical space becomes the essential concern when developing applications in this type of computing environment (Dix et al., 2000). According to Schilit, Context-Aware Computing is about a mobile computing application's competence in discovering and reacting to the changes in the context in a particular situation (Schilit, 1995). This definition has directed the researchers in the field of exploring context support for mobile computing to focus on the context of user's location, nearby people and objects, and changes to the objects. Other supplement to this use of context is the information of time of use (Pascoe et al., 1998). The applications can provide the user with information and functionality that is best suited for her current situation.

Some authors argue that current human computer interaction lacks the ability to communicate and negotiate the rapid changing situation about the interaction with their users. They intend to push human computer interaction closer to interaction between humans to take advantage of common knowledge and implicit understanding. Instead of concentrating on the mobile computing paradigm, other researchers follow a more general approach to Context-Aware Computing. They argued that what people are capable of doing and their experience in certain situations should be included in the computer system design process to achieve Context-Aware Computing (Selker and Burleson, 2000). A Context-Aware System can take advantage of some knowledge of its user's activities when helping her to accomplish tasks. This approach echoes the consensus of Ubicomp and Tangible Bits. Users can concentrate on performing their tasks to achieve a certain goal in a particular situation with the help of computer systems while making the users less aware of technological infrastructure.

To have a more general definition of Context-Aware Computing to suit the approaches mentioned above, we adopt the explanation proposed by Dey et al (Dey, 2000).

> *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves. Therefore, a system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task".*

This thesis will not give new definitions of context and context-aware system. It will focus on a prototyping environment for interface designers to validate claims about context in mobile computing environments. The aim is to provide the designers a reflection of what the context is, where it is found, how it is used in this type of computing environment and discover more about mobile computing device use.

1.3 A TAXONOMY OF CONTEXT-SENSITIVE MOBILE COMPUTING SYSTEMS

There is considerable disagreement over terms such as "context awareness" in human computer interaction (Dey, 2001, Dourish, 2001, Schmidt, 2000). Previous location-aware systems claimed a form of context sensitivity because they used information about the location of the user to make inferences about their likely activities (Davies et al., 1999, Harter et al., 1999, Long et al., 1996, Pascoe et al., 1998, Want et al., 1992). This enabled them to filter information and to offer input options that were intended to support users' predicted tasks. The recent change in focus towards systems that actively sense the physical objects in their environment has led a number of authors to distinguish between those systems and those that simply make inferences based on location information (Cheng and Johnson, 2001). Context awareness implies the ability to directly sense properties of the users' environment rather than simply making predictions based on the users' probable location (Schmidt, 2000). Figure 1.1 illustrates the distinctions mentioned above and also provides a high-level overview of the architectures that will be described in the Chapter 4 of this thesis. This diagram does not use the term "awareness". This is because the word can have the misleading effect of encouraging strong claims about the ability of these systems to consistently provide their users with appropriate information. Current technology supports a very primitive form of "awareness" (Dey, 2000, Schmidt et al., 1998). It should be noted that Figure 1.1 is not intended to provide a complete overview. Further distinctions are likely to emerge with new forms of context-sensitive interaction.

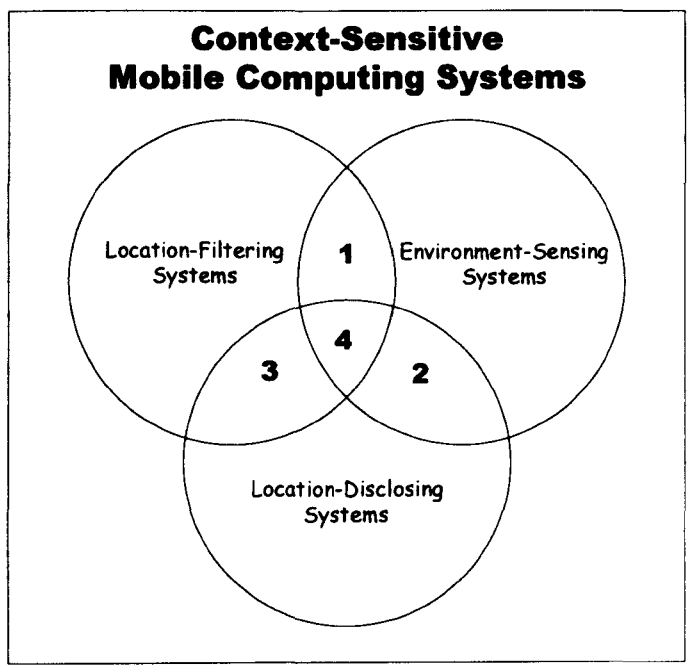


Figure 1.1 The Relationships between Different Types of Context-Sensitive Mobile Computing Systems (Johnson and Cheng, 2002).

We argue that the domain of context-sensitive mobile computing can be divided into at least three types of interactive systems: location-disclosing, location-filtering, and environment-sensing. The first type of context-sensitive mobile computing includes systems that disclose information about the user's location, such as the Active Badge systems mentioned above (Harter and Hopper, 1994). This information can inform subsequent interaction. For instance, a user might prefer to telephone rather than email a colleague who is out of the building. The Active Badge system need not, however, provide any location-dependent information to the person carrying the badge. In contrast, location-filtering systems exploit knowledge of the user's current location to tailor the information that is presented to them. Examples include the Guide (Cheverst et al., 2000) and Cyberguide (Gregory Abowd, 1997) applications. A third type of context-sensitive system actively senses surrounding objects, people and properties of the users' immediate environment. In their pure form, these sensing systems need not have any explicit data about the user's location. For example, short range wireless communication technologies, such as Bluetooth, systems can detect and create links between enabled devices without any location information.

Figure 1.1 also illustrates the intersections between these different approaches. For example, the region 1 represents systems that detect the user's location and other objects in their environment. Systems in this region need not, however, disclose information about the user's location to other people. This approach is embodied in Hewlett Packard's cooltown project (Kindberg and Barton, 2001). The proponents of this form of mobile computing are concerned to avoid the privacy concerns that have affected previous generations of location-disclosing systems. In contrast with systems in region 1, systems that belong to region 2 disclose information about the user's location and sense information about their environment but do not tailor interaction using knowledge of the users' location. This approach is implemented in a number of context-sensitive collaboration works (Kortuem et al., 1999). Most interface designers seem to be eager to exploit information about the user's location and the objects in their environment to inform subsequent interaction (Davies et al., 2001). Systems in the region 3 tailor the presentation of information to their users depending upon their probable location. They would also disclose information about that person's location but would not actively sense other objects in their environment. The ParcTab system is an example of such an application (Want et al., 1995). Finally, section 4 includes hybrid systems that support elements of all of the main context-sensitive systems. In the Chapter 4, we will describe how the Glasgow Context Server (GCS) implements this approach. The GCS was designed to provide a flexible test-bed for the prototyping and evaluation of context-sensitive mobile computing human-computer interfaces.

1.4 PROBLEM STATEMENT

There is not enough knowledge about where to find context information and how to exploit this information in the operation of systems in the mobile computing. Previous approaches in context-sensitive mobile computing are ad hoc in terms of the use of software and hardware (Dey, 2000). It is difficult for interface designers to build a context-sensitive system to validate the claimed benefits of

this type of systems, such as less distraction and input requirements from the users while using the systems (Anhalt et al., 2001, Brown and Jones, 2002, Lieberman and Selker, 2000). In addition, there is little known about the usability issues in context-sensitive systems. Therefore, we propose a prototyping approach to build context-sensitive systems. This prototyping environment has been built in an attempt to help interface designers rapidly build this type of system. In addition, there are demonstrations built upon the prototyping environment as test-beds to validate the claims and point out the usability and social issues in this type of system. These include systems built by a small number of other designers.

This thesis also examines the engineering challenges that interface designers must address if they wish to achieve the benefits proposed for context-sensitive mobile computing. The focus of the thesis is on systems that will work in an indoor environment. This is appropriate interaction because it is the context in which the greatest benefits have been claimed for context-sensitive systems (Harter et al., 1999). It also poses the greatest engineering challenges. Physical barriers prevent GPS from supporting the location detection that is a basic building block for previous context-sensitive mobile computing approach.

1.5 AIMS OF RESEARCH

In this section, the main aims of this research are summarised. Our intention is to present a prototyping environment for context-sensitive mobile computing interaction. The aim is to provide a means of discovering whether or not some of the claimed benefits for this type of interaction could actually be realised by the evaluation of prototype systems. More specifically, the key research aims of this thesis are:

What Is Context-Sensitive Mobile Computing Good for?

Each existing context-sensitive mobile computing system has its own claimed benefits. More detailed discussion on this issue will be described in Chapter 2. In addition, although there are many existing projects for the use of context in mobile computing interfaces there is little known about how to determine the context and where to find it in each different application scenario. The main aims of this part of the research are:

- To categorise the existing context-sensitive mobile computing systems into specific application domains and summarise their claimed benefits.

- To highlight common agreement of what context information should be considered in which application domain.

What Are the Challenges for Existing Context-Sensitive Mobile Computing Systems?

Previous approaches in context-sensitive mobile computing have often been ad hoc. It is difficult for interface designers to build a context-sensitive system to validate many of the claimed benefits. In addition, there is little known about the usability issues in context-sensitive system. The main aims of this part of the work are:

- To examine the existing context-sensitive mobile computing supporting architectures, in particular, their implementations.
- To summarise the limitations of previous work.

How to Address the Challenges from the Existing Approach?

In order to tackle the challenging issues from previous approach, the main aims of this part of work are:

- To build a context-sensitive mobile computing prototyping environment to address some of the limitations of existing approaches.
- To examine the engineering challenges that arise during the development of the environment.
- To evaluate the effect of the prototyping environment. More specifically, this evaluation not only should judge the prototyping environment functionality but also judge whether others can build context-sensitive applications based on the environment.
- To build applications based on the environment as test-beds to validate the claims for context-sensitive mobile computing.

1.6 THESIS OUTLINE

This chapter describes the foundation concepts behind this research. It also presents and discusses the domain of context-sensitive mobile computing interaction. This helps us to draw the boundaries of context-sensitive mobile computing systems when building a prototype system. As we mentioned in the previous section, there is a need for a prototyping environment that supports context-sensitive mobile computing systems. The following sections describe how we implemented this approach and help interface designer to validate claims about this type of system.

Chapter 2 reviews the claimed benefits of context-sensitive mobile computing from previous research in this area. It also surveys the existing research work in different application domains. In particular, it focuses on their application scenarios, use of context information, and underlying technologies. The intention is to identify the claimed benefits from the previous research experiences.

Chapter 3 reviews the implementation of underlying technologies that help the research projects mentioned in the Chapter 2 to obtain context information from their users and environments. This includes analysis of system support infrastructure, sensing technologies that are used to measure the users and environment, and discusses why the existing work is difficult to implement.

Chapter 4 presents the design of the Glasgow Context Server (GCS). The GCS is a prototyping environment for context-sensitive mobile computing interaction based upon off-the-shelf development components. It has been developed to address the concerns from the previous approach discussed in Chapter 3. The aim is to enable designers in this field to rapidly build context-sensitive mobile computing systems.

Chapter 5 describes the practical challenges that arose during the development of the GCS. The reason for this is that our development experiences might act as a blue-print for other interface designers. This chapter also presents the application programming interface (API), with which designers can build context-sensitive mobile computing applications based on the GCS prototyping environment.

Chapter 6 describes the design, implementation, and evaluation of the Virtual Notelet. The Virtual Notelet is a mobile computing annotating application based on the GCS environment. Three primary reasons motivate this presentation. The first is that it demonstrates the design process described in the Appendix B. The second is to illustrate that the GCS environment is working. The third is to exploit the GCS application as a test-bed to validate the claimed benefits of context-sensitive mobile computing as well as discuss the usability and social concerns.

Similarly, Chapter 7 presents the design, implementation, and evaluation of the Shopping Assistant application based on the GCS environment. The motivation for this goes beyond the reasons described for Chapter 6. It also demonstrates that other designers can use the API to build GCS applications.

In Chapter 8, the summary of this research is presented. It also reviews the design of the GCS environment and identifies future work to improve current limitations. This includes more on software architecture support in terms of communication between the users, mobile computing devices, and physical entities. Finally, the conclusions are discussed.

Appendix A describes a design guideline of context-sensitive systems for novice users based on (Faulkner, 1998).

Appendix B presents an outline design process for context-sensitive mobile computing applications. It is not yet a fully articulated design method, but consists of the key concepts and activities that were used during the development of the case studies reported in Chapters 6 and 7. It provides a reference for designers to identify the context information that their applications need in order to exhibit context-sensitive behaviour.

CHAPTER 2
CONTEXT-SENSITIVE MOBILE COMPUTING APPLICATION DOMAINS

In Chapter 1, we noted two reasons for the emergence of context-sensitive mobile computing. Firstly, the advance of mobile computing devices and wireless communication technologies enable users to move around with computing power and network resources at hand. Secondly, unlike traditional stationary computing environments, people carry mobile computing devices while they are on the move. Using applications run on mobile computing devices may cause much distraction and make them difficult to use while the user is engaged in something else. Many of those problems start from a need to provide explicit interaction in operating the device in order to obtain relevant information. Current mobile computing devices are not "context-sensitive". The devices cannot take advantage of context information from their surroundings to reduce need for explicit interaction. As an example, when we talk to a person, we can make use of context information such as gestures and voice tone as implicit input. We are capable of interpreting it and react appropriately. Using context information, computer systems could be made more adaptable. It is especially important in mobile computing environment where the context and user needs change frequently and rapidly. Human-computer interaction in mobile computing systems can be supported if it can obtain and utilize implicit input from their users, environment, and surroundings, and act upon the input along to produce relevant information to their users.

In this chapter, we summarise the claimed benefits from previous approaches in this area and categorise existing research work into specific domains. We analyse the interaction in each different application domain scenario. The aim is to identify the claimed benefits of context-sensitive mobile computing from these works.

2.1 WHAT IS CONTEXT-SENSITIVE MOBILE COMPUTING GOOD FOR? – THE CLAIMED BENEFITS FROM PREVIOUS WORK

Many researchers in this area have claimed the value of context-sensitive mobile computing systems is that they can exploit not only explicit input from their users but also implicit input from the user's current surroundings. Systems can use this to provide relevant information based upon these inputs (Chen and Kotz, 2000, Dey, 2001, Schmidt, 2000). More detailed, context-sensitive mobile computing systems can obtain and exploit implicit inputs to provide their users with tailored information based on the user tasks. According to Schmidt, an implicit input is a user action that the user does not intend to perform to interact with a computer system. However, the system recognises its meaning and

considers as an input (Schmidt, 2000). Implicit inputs may come from sensors, which sense factors about user's activities, surroundings, location, etc. The effect is to reduce the explicit input effort from the users (Anhalt et al., 2001, Selker and Burleson, 2000). According to Lieberman et al, it is important to identify the user, the user-system task, and system models of a context-sensitive system in order to validate how it simplifies the interaction scenario. The user-system task model captures actions that are performed by a user to complete a task with help from a system. User models can hold information about the user's current and past state and preferences. System model means the capabilities of the system itself (Lieberman and Selker, 2000). We are not focusing on user modelling in this thesis as it is a research area in its own right.

We are interested in the analysis of users, user-systems, and systems task in existing context-sensitive mobile computing systems. The following sections describe the related task analysis models and notations that may help us to find out those tasks performed during an interaction.

The following sections focus on related context-sensitive mobile computing systems and their application domains. In particular, we are interested in the analysis of users, user-systems, and systems task in each system. We look at the comparison between context-sensitive and non-context-sensitive approach based on the same scenario to illustrate differences in the resulting effect. This emphasizes the benefits of context-sensitive mobile computing systems. The underlying system infrastructure supports and reasons why existing works are difficult to implement are discussed in Chapter 3.

2.2 TASK ANALYSIS MODELS AND NOTATIONS

A number of researchers have made claims about the benefits of context-sensitive systems. For instance, this type of system can reduce the amount of explicit input as well as attention performed by a user when using the system (Anhalt et al., 2001, Chen and Kotz, 2000, Selker and Burleson, 2000). To validate the claimed benefits, first of all, we must understand what context is considered and how it is exploited in the existing systems. Our approach is to utilize task analysis techniques to analyze the interaction between users and context-sensitive systems to see what and how user's actions could be reduced. The following sections describe a number of task analysis models and notations that could be used to validate the claims.

2.2.1 ConcurTaskTrees

ConcurTaskTrees (CTT) provides a graphical notation to help an interface designer model hierarchically user and system tasks during an interaction. The underlying concept of CTT is that people usually tend to split a problem into small problems yet maintaining the relationships among the small pieces of the solution. The graphical and tree-like representation makes it logically structured

and easy to be interpreted by interface designers. In addition, the temporal relationship support in CTT is able to describe many possible situations, such as concurrency, interruption, disabling, and interaction. Using CTT, designers can concentrate on the most relevant aspects that encompass both user and system-related aspects without low level implementation details when designing interactive systems at early stage. A CTT task model is constructed in three phases. Firstly, the tree-like structure is built using a hierarchical logical decomposition of the tasks. Secondly, the temporal relationships among tasks are identified in each level of that tree. Thirdly, the objects and the actions which allow them to communicate are identified in each level. Consider the following example from (Navarre et al., 2001):

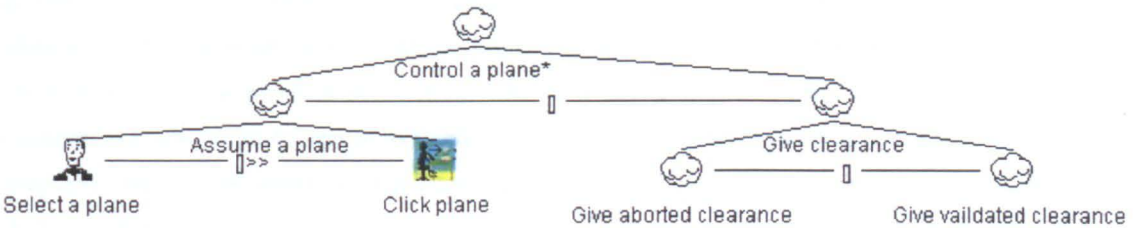


Figure 2.1 ConcurTaskTrees Model (Navarre et al., 2001).

This example demonstrates activities of controlling an airplane. The root task, "control a plane", is indicated as an iterative task (marked by the operator, "*"). It is split into two subtasks: "assume a plane"[1] and "give clearance". The operator, "[]", between these two subtasks indicates that they are mutually exclusive. The task of assuming a plane consists of deciding which plane is selected by the system user and selecting the button related to the plan once the former activity is performed. This is noted by the operator, "[]>>". Similarly, the task of giving clearance consists of two mutually exclusive activities: "give aborted clearance" and "give validated clearance". These two activities are still at an abstract level and cannot be clearly allocated either to the user or the application or the interaction between them. CTT categories these tasks into: abstract, user, application, and interaction tasks.

Abstract tasks, which are displayed with cloud-shape icon, require further analysis and decomposing process to embody user-system actions. User tasks require cognitive or physical activities without interacting with the application, for instance, the task of selecting a plane in this case. Application tasks are completely activated and executed by the application. Interaction tasks require explicit actions in the user interface, for instance, the task of clicking an airplane icon in the example.

CTT provides a rich set of operators and temporal relationships, which are generally easy to understand and used. Task trees are well-suited for communication purposes within the design team.

[1] "Assume" here perhaps means "identify".

Some information could be given about the ordering of tasks, however, no information about roles, actors or objects is given. This will be discussed in Chapter 6 and Chapter 7.

2.2.2 User Action Notation

User Action Notation (UAN) is a user centred and task oriented notation. The notation describes the user-system tasks during an interaction. It was initially designed to help the communication between interface designers and implementers. This technique focuses on the system interface rather than user complexity. The tasks are split into actions in UAN. The decomposition does not stop until primary actions are discovered. Therefore, the highest level in the hierarchy contains all the possible tasks whereas the lowest level contains all the primary actions and the corresponding system feedback. These actions are the physical user actions when interacting with a system. There are a number of elements in UAN: symbols, operators, table of user action, feedback, system action, system state, and temporal relations and constrains. The elements are described in the following example from (Hix and Hartson, 1993):

```
TASK: set alarm
USER ACTIONS                INTERFACE FEEDBACK          INTERFACE STATE
view level = time slot:
  (~[alarm icon]
  Mv                        alarm-icon-!: alarm icon! set alarm mode =
on
~[time slot']               outline(copy(alarm icon))
                            > ~ time slot' !
-------------------------------------------------------------------------
M^)                         @x', y' in time slot'     set alarm mode =
off
                            display(copy(alarm icon))
-------------------------------------------------------------------------
```

The contents within the same horizontal boundary are related to the same action. The temporal sequence is represented by vertical position in respect to the boundaries. The example shows the task, "set alarm", is done by dragging (indicated by "~") a copy of the "alarm icon" to the context of the "time slot", which is selected and indicated by "'". M with up and down arrow indicates the mouse button released and pressed. In the interface feedback column, "!" means highlighting the icon, "alarm icon", whereas "-!" indicates removing the highlight. In the interface state column, it shows the state variable, "alarm mode", changes from "on" to "off" after the user actions. There is only one type of temporal relation, sequence, in this example. UAN also supports to describing other type of temporal representations, such as iteration, choice, repeating choice, order independence, interruptability, interleavability, concurrency, and interval-waiting. This helps describe complex temporal relations between tasks (Hix and Hartson, 1993, Palanque et al., 1995).

The advantage of UAN is that it provides a notation specifically for describing systems with graphical interfaces. The contents in the user actions column in UAN are rather close to the implementation domain. This is easily comprehended by interface developers if they are familiar with the notation. In this chapter, our intention is to have a glance of user and system tasks carried out in the existing context-sensitive mobile computing systems instead of knowing the details about how the actions are implemented. UAN specifications for a complex system could be tedious and hard to read. In addition, it is a non-trivial task to describe non-system supported user activities using UAN. According to Duncan, the natural language of the task environment should be used in the analysis in order to comprehend and have high level view of tasks in a situation rapidly (Duncan, 1972). This approach is more appropriate than UAN when analysing user actions in a non-system supported situation.

2.2.3 GOMS

This technique models some aspects of users' understanding, knowledge, and intentions when interacting with an application interface. A number of characteristics of task analysis techniques, such as CTT and UAN mentioned previously, may be similar to goal-oriented cognitive models. The overall goal is decomposed into subgoals. The decomposition process can carry on repeatedly until some level of detail is discovered. However, the underlying concepts of the two approaches are rather different. Task analysis concentrates more on observing users and their actions whereas a goal-oriented cognitive model is to comprehend the internal cognitive processes when a person performs tasks. GOMS consists of four elements to represent the user's cognitive structure: goals, operators, methods, and selection rules. Consider the following example from (Dix et al., 1998):

```
GOAL: ICONIZE-WINDOW
.       [select GOAL: USE-CLOSE-METHOD
.       .       MOVE-MOUSE-TO-WINDOW-HEADER
.       .       POP-UP-MENU
.       .       CLICK-OVER-CLOSE-OPTION
        GOAL: USE-L7-METHOD
.       .       PRESS-L7-KEY]
```

In this example, the overall goal is to close a window and minimise it to an icon. The dots in the GOMS model are exploited to indicate the hierarchical level of goals. In GOMS, a goal describes the aim which the user intends to achieve and it determines a set of methods needed to achieve that goal. An operator represents actions that the user must perform in order to use the system. Those action executions may affect any aspect of the user's mental state or affect the task environment. However, a GOMS model does not support any temporal relationship of concurrency. A method is the procedure needed to achieve the goal. It can be specified as a conditional sequence of goals and operators, such as the CLOSE-METHOD and L7-METHOD. In order to reach the desired goal, there might be more that one method available to the user. This is where selection rules come into play. In GOMS, method

selection is handled by a set of selection rules. Each selection rule is of the form "if *condition* then *use method*".

A GOMS model is designed to express the internal cognitive processes when a person performs tasks. Thus, system actions as well as feedback are not stated in a GOMS task model. This raises difficulties for us to capture system tasks in system-supported scenarios. In this research, we mainly focus on observable users' actions rather than their internal cognitive state. Therefore, this technique is not suitable for our approach. CCT provides a more complex model that combines user's cognitive goals and computer system actions (Kieras and Polson, 1985); although potentially usable for our analysis, it is perhaps overly complex for our purposes.

2.2.4 Unified Modelling Language

Unified Modelling Language (UML) is arguably one of the most influential modelling languages at present (Pooley and Stevens, 1999). It provides standardised models that were used in Object Oriented Analysis and Design. UML is represented diagrammatically. It was not initially designed for task analysis. However, several diagrams can be utilized in performing a task analysis. To carry out this approach the interpretation in UML needs to be altered slightly. For example, states in an activity model are considered as tasks. UML has four representations that are directly relevant for task modelling: activity diagram, collaboration diagram, sequence diagram, and use case diagram. The activity diagram can be used to describe the task flow in relation to events, user roles and goals. The collaboration diagram provides an overall view of the way different objects work together. The sequence diagram can illustrate the sequence of tasks performed between different user roles. The use case diagram can be exploited to describe interaction scenarios. We consider use case diagram could be useful for task analysis. A use case describes a specific "path" through a task tree under predefined conditions.

The potential benefit of using UML diagrams is that software engineers might be familiar with them. Many software tools exist that support the designers to create UML diagrams. However, other disciplines may not know them. Also, the diagrams would require a small adaptation for task modeling purposes.

2.2.5 Hierarchical Task Analysis

Hierarchical Task Analysis (HTA) focuses on the way a task is decomposed into subtasks and the order and conditions where these are executed (Dix et al., 1998). They are represented as a hierarchy of tasks, subtasks and plans. The output of HTA can be represented textually and diagrammatically (Dix et al., 1998, Shepherd, 1989). Consider the following example from (Dix et al., 1998):

```
0. in order to clean the house
        1. get the vacuum cleaner out
        2. fix the appropriate attachment
        3. clean the rooms
                3.1. clean the hall
                3.2. clean the living rooms
                3.3. clean the bedrooms
        4. empty the dust bag
        5. put the vacuum cleaner and tools away

Plan 0: do 1-2-3-5 in that order.
        when the dust bag gets full do 4

Plan 3: do any of 3.1, 3.2, or 3.3 in any order
        depending on which rooms need cleaning
```

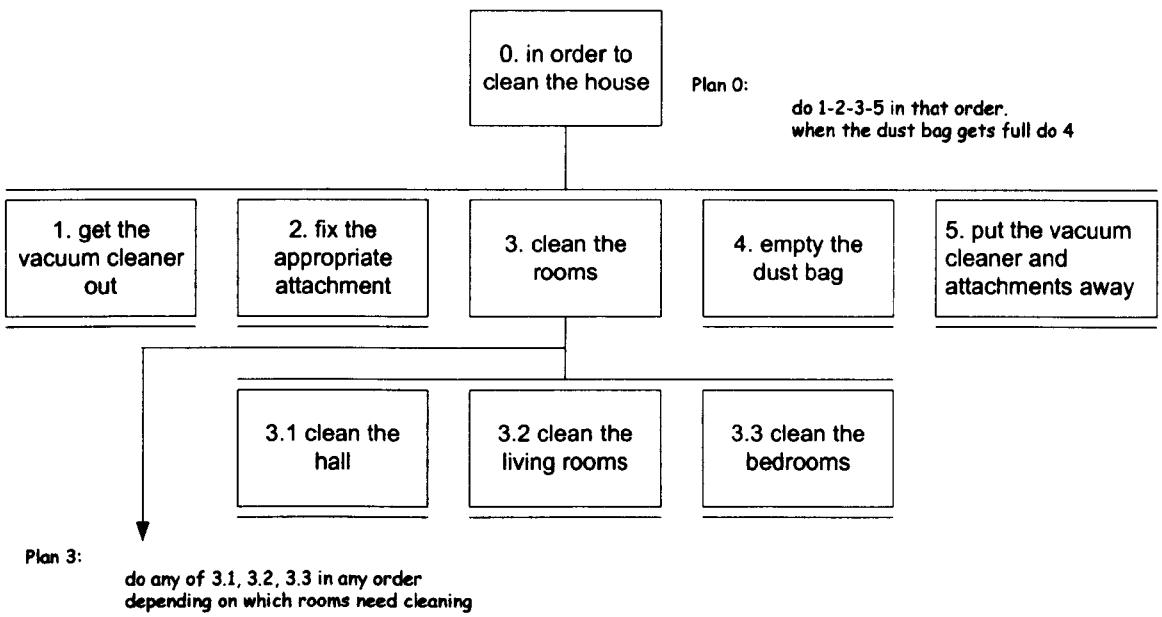


Figure 2.2 Diagrammatical Representation of HTA (Dix et al., 1998).

The example is presented in both textual and diagrammatical forms. In the example, the task, "clean the house", is split into several subtasks. Each subtask is described in terms of its objective. Some subtasks can be decomposed further, such as subtask 3. The HTA shown textually above are numbered and indented to indicate this hierarchy. In diagrammatical form of HTA, each box is numbered and represents a task or a subtask. The underlining of boxes is exploited to indicate that there is no further task decomposition offered. In both representations, the overall task is numbered 0 whereas subtasks are numbered from 1 to whatever is necessary. The numbering also shows task hierarchy, such as subtasks 3.1 to 3.3 follows subtask 3 in this case. The plans describe how these tasks are performed and are marked by the corresponding task. It shows the sequence of subtasks and

event waiting (subtask 4 in this case). It can also present cycles (doing a certain task or group of tasks repeatedly until a particular condition is reached), time-sharing (two or more subtasks to be carried out at the same time), discretionary (doing any one of subtasks without constrain, such as plan 3 in this case), and mixed (mixture of any form of plans mentioned) plans.

However, this is not the only HTA representation of the house cleaning task. For example, the subtask 4, "empty the dust bag", could be performed after subtask 1 or 5. The plan 3 could add schedule of house cleaning to improve the performance. In addition, it is necessary to ensure a person is capable of carrying out each task and subtask in the HTA. If there is any task or subtask that a person or a system does not possess enough competence to perform operations at the task, further task decomposition process is needed to address this problem. Restructuring, tailoring, and finding the suitable and meaningful hierarchy for a particular situation is part of the process of HTA. Further decomposing tasks into subtasks and redescribing them into plans consumes much more time and may not improve the overall performance if unnecessary task decomposition is performed. The P × C Stopping rule is proposed to address this issue. The idea of this rule is that if the result of probability of making a mistake in the task, P, multiplied by the cost of the mistake, C, below a threshold, the decomposition process stops (Annett and Duncan, 1967, Dix et al., 1998, Shepherd, 1989).

HTA has the potential to represent goal structures and cognitive decision-making activities. As mentioned previously, the decompositional, sequence representing, and task operation selecting nature of GOMS is similar to the HTA approach. Some researchers argue for the advantage of HTA is at an early stage in the design process (Carey et al., 1989). It can provide brief picture of user tasks and basic functional specification of the proposed system. It can also be exploited as a rough-and-ready tool at this stage. The decompositional nature enables analyst to concentrate on parts of the overall task without losing the picture of overall task activities. In addition, the top down structure ensures completeness and is easy to comprehend (Carey et al., 1989, Shepherd, 1989).

In this research, we intend to exploit existing task analysis techniques instead of creating a new approach to validate existing context-sensitive mobile computing systems. HTA provides an easy entry level for us to inspect existing context-sensitive mobile computing systems. Regarding the task-design mapping, HTA provides a clear description of all task functions for mapping on to the new interactive system. It is also ideal for the identification and mapping of information input and output requirements in system design (Carey et al., 1989). We can take advantage of this easy yet hands-on task analysis technique to identify the input and output of a context-sensitive system in order to understand how the developers of this type of systems consider context information and how they transform the information as input to their systems. The textual representation of HTA in this chapter and the rest of this thesis follows the format of the aforementioned example, which is listed in (Dix et al., 1998).

The following sections focus on related context-sensitive mobile computing systems and their application domains. In particular, we are interested in the analysis of users, user-systems, and

systems task in each system. We look at the comparison between context-sensitive and non-context-sensitive approach based on the same scenario to illustrate differences in the resulting effect. This emphasizes the benefits of context-sensitive mobile computing systems. The underlying system infrastructure supports and reason why existing works are difficult to implement is discussed in the Chapter 3.

2.3 APPLICATION DOMAINS

In the following section, we look at existing work in context-sensitive mobile computing. The aim is to extract the essence of context-sensitivity as benefit to human-computer interaction. We categorise the research work into different application domains such as office utilities, tour guides, shopping assistants, social utilities, games and field work assistants. In particular, we examine how they use context information to simplify their users' tasks. It is important to derive the transformation of user, user-system, and system task from each application domain. Non-system support scenarios are shown as a contrast for each application. This serves to elicit the benefits of context-sensitive support for mobile computing systems. Task analysis tends to focus on the observable behaviour of users when they perform tasks in certain situations (Dix et al., 1998). Hierarchical task analysis (HTA) is used to describe the interaction that takes part in performing tasks both with and without system support. The intention for this is to identify context information from the task and interaction flows. We chose this task analysis technique for pragmatic reasons and further studies could compare this approach with alternatives such as Use Case (Tuulari, 2000). The context-sensitive infrastructure mentioned in each application domain is described in terms of what and how to obtain context information and the underlying hardware and software design in the Chapter 3.

2.3.1 OFFICE UTILITIES

The early demonstration of context-sensitive mobile computing systems focused on tailoring information and presenting it to users based on the current situations in the office domain. The intention is to improve not only the interaction between office workers but also the effectiveness of using computers, printers, and electronic equipment that helps with tedious repetitive tasks, such as communication, security checks, etc. in the office domain. Existing context-sensitive applications include in-out boards, call-forwarding, teleporting, which is also known as follow-me computing, and resource discovery (i.e. people and equipment).

2.3.1.1 IN-OUT BOARD

The "In-out" board displays the in/out status of colleagues in an organization. It usually appears around the main entrance or common area in an organization. This information can be used by co-workers to inform subsequent tasks, such as the call-forwarding discussed in the following section. It

is an implicit consensus that when people leave or enter their working places, they should indicate their in/out status in order to support interaction with other colleagues in terms of collaborative work and social activities. The following shows the HTA of adjusting one's status on an in-out board both with and without system support:

Figure 2.3 In-out Board

Non-System Support:

```
0. adjust one's in/out status on an in-out board
        1. go to the place where the in-out board is located (User task)
        2. find one's name on the board (User task)
        3. move the maker to indicate in/out status (User task)
Plan 0: do 1 - 3 in that order

0. check someone's in/out status
        1. go to the place where the in-out board is located (User task)
        2. find the name on the board (User task)
        3. look at in/out status

Plan 0: do 1 - 3 in that order
```

We refined the HTA and include iButtons (described in the following section) in the analysis to see what happen when users use this technology.

System Support:

```
0. adjust one's in/out status on an in-out board
        1. leave one's working place (User task)
        2. go to the place where the sensor reader is located (User task)
        3. dock the sensor on the reader (User task)

Plan 0: do 1 - 3 in that order
0. check someone's in/out status
        1. link to the In/Out Board application using a web browser (User-
        system task)
        2. find the name on the board (User-system task)

Plan 0: do 1 - 2 in that order
```

The scenario of adjusting one's in/out status with system support is based on the Georgia Tech's In/Out Board application (Salber et al., 1999). Using this system, users are required to carry iButtons[2]. The users need to attach their iButtons, which with unique ID, to the correspondent reader located at a fixed place. The designers consider the context information in this system is location, user's identity, and time (Dey, 2000). The HTA reveals that only one action, task 3 to adjust one's in/out status without system support, can be simplified by the system and its underlying infrastructure when the user adjusts his/her in/out status. With the system support, the user still has to go to the place where the sensor reader situated and explicitly docks the iButton on it, in order to activate the change. The task of finding the user's name on the board becomes obsolete because the system can obtain the user's identity from the sensor. In checking other colleague's in/out status, the system provide the users with a remote access option through web technology. The users simply link to the in/out board application web page and other colleagues' in/out status is then showed on the page. This reduces the user's effort to go to the in/out board in person.

This illustrates key point about HTA gives no idea of user perception of effort involved in each step. Ideally, the system should detect the user's location to determine whether s/he is in the office or not. It can be argued that the system should sense the user's current location. There should be no need for the user to do the sensor docking task in person in the first place. The user can simply focus on his or her current task. This type of application utilizes the user's location information, his/her identity, and timestamp of last seen as context to help automatically change the user's in/out status to simplify the user's task. It is argued that the users want to have more control over the subsequent interaction tasks, depending on their current situation such as they do not want to take unexpected calls or receive instant messages when they are in a meeting (Adams, 2002, DeVaul and Dunn, 2001, Schigeoka, 2002). User's preference should be taken into account to make the application meet their users' social need. This is described in the call-forwarding and social utilities section.

---

[2] http://www.ibutton.com/

2.3.1.2 CALL-FORWARDING

The call-forwarding application scenario is that the users can be tracked within a building and phone calls are forwarded to the nearest phone to them. The first demonstrations were based on the Active Badge system (Want et al., 1992). Users are required to wear badges and move around the building. Their location information can be obtained and updated to the database by the system. The database contains information about the users' current or most recent location, whether or not they are in their working places or offices. It also contains status message, and the nearest phone extension. When the receptionist receives a phone call for a particular user, she can use the database to look up the recipient's location information and forward the phone call to his/her last known location. The following scenario is based on the call-forwarding application at AT&T Laboratories Cambridge and assumes that the intended recipient exists (Cambridge, 1992a) (Cambridge, 1992b). This is not the only HTA we can produce. It is simply an example. The HTA of call-forwarding is as follows:

Non-System Support:

```
Receptionist:

0. Receptionist forwards calls to intended recipient

        1. pick up the incoming call (User task)
        2. converse with the caller (User task)
        3. identify the intended recipient (User task)
        4. check the recipient's status (User task)
                4.1 check the recipient's in/out status from in-out board
                (User task)
                4.2 check the recipient's schedule (User task)
        5. appoint the next call with the caller for the recipient (User task)
        6. forward the message to the recipient (User task)
        7. check the recipient's extension number (User task)
                7.1 Get the phone list (User task)
                7.2 Look up the phone list to obtain the recipient's extension
                number (User task)
        8. forward the call to the phone in the office or somewhere close to
        (User task)

Plan 0: 1 - 4 in that order
         if the recipient is not available
         then 5 - 6
         else 7 - 8

Plan 1: do 4.1 - 4.2 in that order
```

Plan 2: do 7.1 - 7.2 in that order

Recipient:

0. update the current status (in/out, in a meeting, and etc.)
    1. go to reception desk/office (User task)
    2. inform the receptionist about in/out status and activity status (User task)

Plan 0: do 1 - 2 in that order

0. receive calls from others

    1. answer the nearest phone (User task)

Plan 0: do 1

System Support:

Receptionist:

0. receptionist forwards calls to intended recipient
    1. pick up the incoming call (User task)
    2. converse with the caller (User task)
    3. identify the intended recipient (User task)
    4. check the recipient's status from application database (User-system task)
    5. appoint the next call with the caller for the recipient (User task)
    6. update the message to the recipient correspondent database entry (User-system task)
    7. forward the call to the phone close to the recipient (Use the result from task 4) (User-system task)

Plan 0: do 1 - 4 in that order
    if the recipient is not available
    then 5 - 6
    else 7

Recipient:

0. update current status
    1. operate the Active Badge (User-system task)

Plan 0: do 1

```
0. Receive calls from others
      1. answer the nearest phone (User task)

Plan 0: do 1
```

The HTA reveals that the application reduces the number of acts required for the receptionist and the recipient to perform call-forwarding. The application considers its users' location and their current status as context information. The system can obtain the context information about the user's identity, location, and timestamp of last seen and update the database automatically. That means the system considers the recipients' movement as an implicit input. This makes the interaction flows of completing the task smoother than doing the task without system support. It reduces the physical activities required to explicitly update information about the recipients' current situation. Updating one's current location requires that the telephone recipients go to inform the receptionist. Also, receptionist need not iterate to check recipients' current status and extension number using paper based list. Instead, the application can help integrate the information about the last known location, status, and closest phone extension to the recipients.

2.3.1.3 TELEPORTING SYSTEM

The idea of follow-me application is to make information related to a particular user "follow" her wherever she goes. The previous example of call-forwarding application falls into this category. Teleporting is another example, which allows users to bring their computing state with them as they move between computers (Cambridge, 1994). The necessary underlying network mechanisms are called Virtual Network Computing (VNC) (Cambridge, 1998b). For example, a user is interrupted and asked to do something in another room while she is writing an email. She can leave her desk, go to another machine, and the system can transfer her previous computing state to the machine so that she can finish the email. This application can also be based on the Active Badge infrastructure to locate users and computers. The following HTA uses Active Bat, which exploits a hybrid approach of ultrasound and radio frequency. The scenario of resuming email writing and sending it out from another machine is presented using HTA as follows:

Non-System Support:

```
0. resume email writing and send it out from one to another machine
      1. stop writing the email (User task)
      2. save the draft on a storage medium (i.e. floppy disk)
      (User-system task)
      3. bring the disk on the way to another place (User task)
      4. insert the disk in the machine you pick up (User task)
      5. execute an email authoring program (User-system task)
      6. load the saved draft from the disk (User-system task)
      7. continue the email writing (User-system Task)
      8. send the email out once you finish the writing (User-system task)

Plan 0: do 1 - 8 in that order
```

System Support:

```
0. resume email writing and send it out from one to another machine
      1. stop writing the email (User task)
      2. leave the current location for another (User task)
      3. pick one machine in the current location (User task)
      4. continue the email writing (User-system task)
      5. send the email out once you finish the writing
      (User-system task)

Plan 0: do 1 - 5 in that order
```

The HTA reveals that the task of transferring the user's computing state from one workstation to another is taken over by the system. The system uses the user's location and the location of the machine (i.e. workstation) as context information to simplify the user's task. Once the system identifies the user and the nearby workstation, the desktop environment on the user's previous operating computer can be transferred to the current workstation. The system dynamically maps the user interface onto the resources of surrounding computer and communication equipment. The user's computing state can go with the user and resume at any computing resources hosted by the system. This approach is utilized to assist to support to resource limited mobile computing devices with surrounding computing resources (Cambridge, 1998b). However, some current application software and other services may not be available at all locations.

2.3.1.4 RESOURCE DISCOVERY

Resource discovery applications can provide their users with information about the nearby resources such as printers, projectors, etc. ParcTab is arguably the first demonstration of this type of application (Want et al., 1995). Similar to the "follow-me" applications mentioned previously, the system can take advantage of information about its user's current location and provide an information redirection service to its users. However, "follow-me" applications make active approach redirect the relevant information from one computer to another while resource discovery applications provide an available resource list to their users. Here is a scenario based on the RADAR system demonstration video (Research, 2000): Assume that a person who will give a presentation arrives 5 minutes before the meeting starts. He realises that he has forgotten to bring the printed handouts. He then uses the resource discovery application to assist him to print out the handouts. The application can inform the person of the location of nearby printers. The user can choose a printer suggested by the system and print out the document. The system can inform the user when and where to collect these handouts.

Non-System Support:

```
0. print out documents from the nearest printer
      1. connect your computer to the network (User task)
      2. search the closest printer from the network (User-system task)
```

```
        3. install the printer driver on the computer (User-system task)
        4. send print out command from the computer (User-system task)
        5. wait for the notification of finishing printing out (User-system
        task)
        6. go to the place where the printer located (User task)
        7. collect the document (User task)

Plan 0: do 1 - 7 in that order
```

System Support:

```
0. print out documents from the nearest printer
        1. connect your computer to the network (User task)
        2. send print out command from the computer (User-system task)
        3. wait for the notification of finishing printing out the document
        and where to collect it (User-system task)
        4. go to the place where the printer located (User task)
        5. collect the document (User task)

Plan 0: do 1 - 5 in that order
```

The HTA shows that the tasks of finding nearby printers and connecting them to the user's computer are transferred to the system. The system utilizes the user's current location and the surrounding resources as context information to simplify the user's task. The user simply concentrates on his task, prints out the document, and need not spend effort to find out where the nearest printer is and install the relevant driver. Similar to the "follow-me" scenario, the user can send the print out command from a computer in his office or home and collects the document in the conference room where the meeting is taking place. Notice that how HTA abstracts away from security and access concerns. As we shall see, this identifies the need for prototyping environment to provide additional features about these concerns.

2.3.2 TOUR GUIDES

When a person visits a city or an exhibition, he can go to an information centre or counter to get a paper-based map and use it to guide himself. However, visitors might get lost if they cannot find the link between the physical place and the map or want to have personalized visiting routes. Context-sensitive mobile computing applications in this domain tend to provide their users with information about their current location and suggest routes based on user's preferences. (Chan, 2001b, Davies et al., 2001, Long et al., 1996, MacColl et al., 2002, Oppermann and Specht, 1999, Spasojevic and Kindberg, 2001, Youll et al., 2000). The user's preference may be obtained from a history of where previous users have been or the user's interests (Galani and Chalmers, 2002). Existing work in this

area can be categorized into either indoor or outdoor system. The difference is their underlying sensing technology. Details are described in the next chapter.

Many indoor exhibitions, for instance, museums provide their visitor not only with paper-based guides but also tape recoded guides. Both mediums provide predefined visiting routes and lack flexibility to adjust themselves to suit their users' needs based on their current situation. For example, a visitor may feel bored with her current route or attracted by a particular exhibit. They may want to have another choice of path. The paper-based and audio guide cannot support the dynamic nature of visitors' interests. The Hippie system avoids this limitation. It is an indoor museum guide using context-sensitive mobile computing support (Broadbent and Marti, 1997, Oppermann and Specht, 2000). The visitor carries a PDA and wears earphones while s/he walks within the museum. Each exhibit is equipped with an infrared transmitter, which is used as a link to the corresponding digital information stored in the system. In the original prototype, information about the exhibits in the museum was cached on a PDA. The current development of Hippie has incorporated wireless LAN to provide dynamic information to the users[3].

Non-System Support:

```
0. visit a museum using paper-based guide
      1. obtain a paper-based guide from the counter (User task)
      2. choose a categorized visiting path (User task)
      3. follow the categorized visiting path (User task)
      4. walk abound the museum (User task)
      5. stop at the interested exhibit (User task)
      6. look up information about the exhibit on the guide (User task)
      7. read the description of the exhibit displayed around the exhibit
      (User task)
Plan 0: do 1
        if categorized visiting route provided on the guide
        then do 2 - 7
        else do 4 - 7

0. visit a museum using audio guide
      1. obtain a audio guide from the counter (User task)
      2. choose a categorized visiting path (User-system task)
      3. follow the categorized visiting path (User task)
      4. walk abound the museum (User task)
      5. stop at the interested exhibit (User task)
      6. press the number displayed on the exhibit on the audio guide
      keypad (User-system task)
```

[3] http://www.fit.fraunhofer.de/projekte/hips/index_en.xml?aspect=guide and http://www.fit.fraunhofer.de/projekte/hips/index_en.xml?aspect=prototype

```
        7. hear the description of the exhibit displayed around the exhibit
        (User task)

Plan 0: do 1
          if categorized visiting route provided on the audio guide
          then do 2 - 7
          else do 4 - 7
```

System Support:

```
0. visit a museum using context-sensitive mobile computing system (Hippie)
        1. obtain the device from the counter (User task)
        2. choose a preferred visiting path organized by the system
        (User-system task)
        3. follow the visiting path (User task)
        4. stop at the exhibit interest you (User task)
        5. information about the exhibit is presented through the earphone
        (System task)
        6. want to discover different topic (User task)
        7. change current visiting path to another (System task)
        8. follow the visiting suggested by the system (User-system task)

Plan 0: do 1 - 4 in that order
          if the visitor is attracted by something else
          then do 5 - 8 - 3 - 4
```

The Hippie development team claim that the application utilizes the user's location/presence and preference as context information to simplify the user's visiting task. From the HTA, we see task 2 and 3 in the non-system support section are conditional while they are unconditional tasks in the context-sensitive mobile computing system support. This emphasises that a personalized visiting path is an essential function for a context-sensitive guide application. To personalize a visiting route for the visitor, the system asks the visitor what kind of tour they would prefer and then guides the visitor based on his/her preference. At this stage, the system needs to gather context about its user's preference explicitly from the visitor. During the visit, the system shows the visitor his/her current location on the PDA and the path to the next planed exhibit. This reduces the effort that the visitor needs to iterate between the paper-based guide and checking the current location. An audio guide is arguably better than a paper-based guide because the visitor can visually focus on the physical environment and audibly receive the direction guide to the next exhibit. The HTA revels that the user's task of finding a description of an exhibit in the non-system support section can be transformed to a system task by adapting to user's location information. The task 5 to 7 in the non-system support section requires explicit interaction between the visitor and a paper-based guide or an audio guide. The visitor has to match the label on the exhibit with either the label on the paper-based guide or press the corresponding label (number) on the audio guide keypad to read or hear the description. However,

the context-sensitive system support can detect the visitor's location and provide him/her the information about exhibits automatically. As for task 6 to 8 in the system support section, it is an exclusive for the context-sensitive mobile computing guide system. For example, the visitor may be interested in a specific exhibit and want to know more about it by selecting the detail information option on the PDA's screen. The system can sense the implicit changes in the visitor's interest from the interaction between the visitor and the PDA. It may then suggest a new route.

Figure 2.4 Museum Audio Guide (Source: http://www.ophrys.net)

2.3.3 SHOPPING ASSISTANCE

The aim of this type of applications is to provide their customers with a pleasant, convenient, and efficient shopping experience. The intention is to address the frustrations that most customers encounter: to find items, to make price comparison, and to improve an impersonalized shopping service. To do this, existing context-sensitive mobile computing shopping assistance applications provide their users with a "personalized shopping experience" based on their needs. Item price, location, discount, etc. are taken into account in this type of applications (Asthana et al., 1994, Chan, 2001a). Some supermarkets offer memberships to their customers. Customers may get special discount on items in the shop once they sign up to become members. This customer profile can be used as base information for the shop to do recommending sales promotion, direct marketing strategy, etc. The following system support scenario is based on the Personal Shopping Assistant (PSA), which is an indoor context-sensitive shopping assistance tool developed at AT&T Bell Laboratories (Asthana et al., 1994). Using this system, each shopper carries a wireless hand-held device while s/he is walking around within the shop. The system can locate the shopper and help him/her to find items in the store. It also exploits voice recognition to simplify the user task while shoppers are walking around.

Non-System Support:

```
0. shop in a supermarket
        1. grab a basket or a trolley (User task)
        2. looking for items on the shopping list (User task)
        3. check the promotion in the shop (User task)
        4. check out (User task)
```

```
        4.1 show the membership card to cashier (User task)
        4.2 wait (cashier may need to scan the membership card)
        (User task)
        4.3 get the card back from the cashier (User task)

Plan 0: do 1 - 4 in that order

Plan 1: if have membership card
        do 4.1 - 4.3 in that order
```

System Support:

```
0. shop in a supermarket using PSA
      1. grab a basket or trolley (User task)
      2. obtain PSA device from the counter in the shop (User task)
      3. "ask" the PSA about the location of each item on the shopping list
      (User-system task)
      4. scan the bar code on each bought item using bar code scanner
      equipped with the PSA (User task)
      5. follow the direction told by the PSA (User-system task)
      6. check out (User task)

Plan 0: do 1 - 6 in that order
```

The HTA shows that task 2 in the non-system support scenario might be expanded into further sub-tasks. For example, the shopper may not find the items on his/her shopping list and may have to ask for help from the staff. In the case of the paper-based shop guide, which provides a layout of the store and promotion items, customers may encounter the distraction of checking the guide while on the move. The PSA's use of context information about its user's location within the store can avoid some of these potential distractions. Also, the AT&T PSA's hands-free approach reduces the interaction demands between the shopper and PSA. The resulting effect is that the shopper can focus on his/her physical shopping environment and benefit from the system help. Task 4 and its sub-tasks in the non-system support scenario are transferred to system tasks in the system support. The system maintains customer profiles and therefore need not ask the customers to identify themselves while paying at cashiers. Normally, customers have to put each bought item on the conveyer and a cashier scans them to calculate the total price. However, the PSA can record the price of each item bought as seen in task 4 in the system support. Shop staff need not scan each item for customers and this keeps the service flowing. Other research approaches provide shopping list support while the user is near the store (Marmasse and Schmandt, 2000, Schmandt et al., 2000). The users can write down their shopping list at home using a conventional desktop PC. The shopping list information is then associated with a corresponding location in the store. Once the user is near the store, the location information is delivered to the user's mobile computing device.

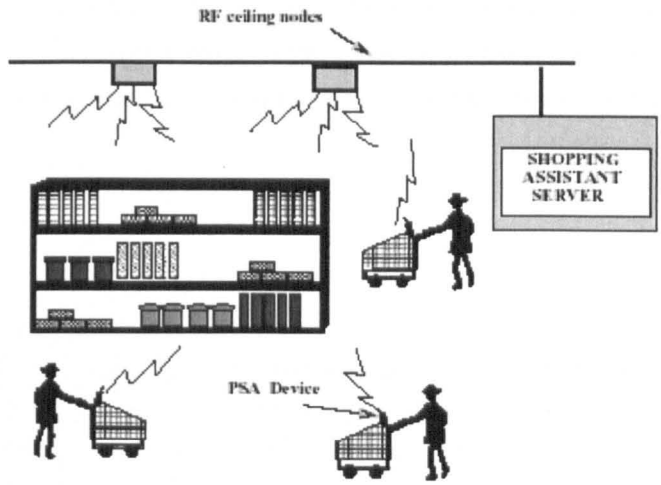


Figure 2.5 AT&T Personal Shopping Assistance (Source: (Asthana et al., 1994))

2.3.4 SOCIAL ENHANCEMENT

The application domains mentioned in previous sections all have different social concerns. For example, people show consideration about their privacy in the office domain. Applications in the tour guide domain can provide users with access to information about the traces of other users, as part of a recommending service. In the shopping assistance domain, the application can inform the shopper that an item was unavailable on the previous shopping is now in stock. This kind of information can influence a user's current task or inform him subsequent tasks. In this section, we focus on the application that provides its users with social communication channel with others using virtual notes. The users can, for example, share their experience on particular things or places with other people. We also look at smart devices that can recognize and adapt themselves to their user's current social situation while providing services.

2.3.4.1 Virtual Notes

The idea of this type of applications is to mimic the existing physical annotation techniques such as Post-It notes, which enable people to put reminders on physical entities (Burrell and Gay, 2002, Pascoe, 1997). GeoNotes provides an example application in this category. GeoNotes helps users to write anonymous notes, as well as comment, filter, and browse virtual notes. These virtual notes can be attached to the user's current location. Using this system, the users carry PDAs equipped with a GPS positioning system while they are walking in the street. The authors argued that people can use GeoNotes as a tool to express their needs (Espinoza et al., 2001). For example, if a user wants to sell his car then he leaves a virtual note at the place where his car parked. Anyone who is interested in purchasing a car can be notified when passing by. In this case, the seller express as his intention about selling his car at a certain price and the potential buyers set their interest in buying a car using a filter

function provided by the system. In the recommending service theme, the users can make comments about dishes and service in a restaurant through virtual notes. Other users can view these notes as suggestions before they go into the restaurant. The following shows the HTA of the restaurant scenario.

Non-system Support:

```
0. look for a restaurant for dinner
       1. look around the surrounding restaurant (User task)
       2. stop by an interested restaurant (User task)
       3. check menu (optional: price) (User task)
       4. check another restaurant (User task)
       5. go into the restaurant (User task)

Plan 0: do 1 - 3
        if  do not satisfied
        then do 4
        else do 5
```

System Support:

```
0. look for a restaurant for dinner using GeoNotes
       1. set the criteria of interest using the filter function in the
       GeoNotes (User-system task)
       2. stop by once the system alert (System task)
       3. go into the restaurant (User task)

Plan 0: do 1 - 3 in that order
```

GeoNotes exploits context information about user's location and preferences to simplify user's task in this scenario. From the HTA, we see the "tedious" tasks of finding a suitable restaurant in the non-system support section can be transferred to user-system and system tasks using GeoNotes. Some authors have argued, however, that users do not pay attention to mobile computing devices most of the time (Anhalt et al., 2001). They usually put the devices in their rucksacks or pockets. The GeoNotes system provides the users with a function, which filters out virtual notes that the users is not interested in. Sound alert is presented once a relevant virtual note is received. This approach allows the user to focus on his/her ultimate goal, for example, have dinner in a preferred restaurant, while walking on the street. The system takes the responsibility for telling the user which restaurant s/he should go to once the context matches – passing by a restaurant that meets the user's requirement.

2.3.4.2 SMART MOBILE PHONE

Current development of mobile phone is not design for context-sensitive. Mobile phone users must set an appropriate operation mode for their social setting. However, users often forget to setup their mobile phone to meet the current situation. Research on context-sensitive mobile phone focuses on the user's current situation, for example, location, activity, and co-location of the user and his/her mobile phone (i.e. in the pocket, in the user's hand, on the desk, etc.) and utilizes the information to enhance the quality of usage in terms of social aspects (DeVaul and Dunn, 2001, Lijungstrand, 2001, Schmidt et al., 2000, Tuulari, 2000). For example, a mobile phone detects that its user is in a meeting and does not want to receive any call except emergency ones. The mobile phone can then adjust itself to the "meeting" mode and apply the appropriate call filter during the meeting. Inspired by instant messaging (IM) services, Schmidt et al implemented their concept of "context-call" over the Wireless Application Protocol (WAP). In this case, the user or the mobile phone itself can publish the current situation and contact method to the central server. Callers contact the user by making a context-call in the same way as using IM services to see the status of a recipient and decide to make a call, leave a message, or call the user later.

The following scenario is based on the context-call development in TecO. We assume the mobile phone in the scenario is context-sensitive as the prototype developed in MIThril. The scenario shows that a person is in the middle of a meeting at the customer site. One of her colleagues is calling her about going for a drink later.

Non-system Support:

```
The person in a meeting:

0. change the mobile phone status to "meeting" mode
       1. press appropriate key set on the mobile phone
       (User-system task)
       2. check/answer the phone (User-system task)

Plan 0: do 1
        if the incoming call goes through the filter
        do 2
```

System Support:

```
The person in a meeting:

0. change the mobile phone status to "meeting" mode
       1. check/answer the phone (User-system task)

Plan 0: if the incoming call goes through the filter
        do 1
```

The HTA shows the task 1 in non-system support section is transferred from explicit user-system task to system task. The user's activity of walking into the meeting room is considered as implicit input for the system. The context-sensitive system support approach allows the user focus on his current tasks in the meeting with his customer and do not have to explicitly adjust his mobile phone to "meeting" mode. The user need not worry about whether his phone has been set to an appropriate mode.

2.3.5 GAMES

A number of recent research implementations have built context-sensitive mobile computing games to expand the arena from virtual space to mixtures of virtual and physical space (Bjork et al., 2001, Falk, 2001, Headon and Curwen, 2001). The aim is to evaluate how traditional game design can benefit from mobile computing, wireless communication, and sensor technologies. They want to investigate how to maintain and encourage social interaction in play. We look at "Pirates!", a context-sensitive mobile computing multi-player game, and apply HTA to illustrate the differences between context-sensitive system support and traditional game playing. This game exploits context information about its player's location, other players' location, and the location of game objects, such as treasure. The game scenario is that each player represents the captain of their ship. They have to walk around the physical game arena to obtain treasure and earn points. They may, however, be engaged in a battle with other ships nearby. Playing this game, the player carries a hand-held device equipped with a wireless connection and a sensor receiver while they are moving around the physical game environment.

Non-system Support:

```
0. play the Pirates! game
       1. move the game character around using game pad or keyboard (User
       task)
       2. search for treasures (User task)
       3. attack other ships (User task)
Plan 0: do 1 - 2 in that order
         if encounter other ships
         then do 3
```

System Support:

```
0. play the Pirates! game using context-sensitive mobile computing device
       1. move around the game character by physically walking around the
       physical game arena (User task)
       2. search for treasures (User task)
       3. attack other ships (User task)

Plan 0: do 1 - 2 in that order
         if encounter other ships
         then do 3
```

From the HTA, we see the task 1 in the non-system support section can be transferred from explicit user task to implicit user task. Namely, the game system regards the player's movement is an implicit input. The game character, the ship, moves while the players walk around instead of pressing the buttons on a game pad. The benefit of system support is that the player can immerse into the game. The immersive experience in the game play would increase the level of excitement when the player playing the game (Headon and Curwen, 2001, Schneider and Kortuem, 2001). Augmented Reality (AR), which tackles the research issue of interaction between human, physical, and virtual entities, is rather suitable to describe the interaction between the player, game application, and physical and digital game arena. Many researchers in this field tend to exploit the context-sensitive game applications as social interaction test-bed to discover more about how the players react to each other on particular game tasks (Dennis, 2001, Pering, 2001, Schneider and Kortuem, 2001).

2.3.6 FIELD WORK ASSISTANTS

This type of applications concentrates on providing utilities to help in fieldworker's observation and data-collection activities (Kortuem et al., 1999, Pascoe et al., 1998). For instance, fixing computer network problems is a tedious task. Sometimes the use of manuals is inevitable. This can cause burdens in checking the manuals against physical "realities" such as indicator light, wires, and traffic data. We look at the NETMAN context-sensitive field work support system to see how it can simplify complex network maintenance tasks (Kortuem et al., 1999). The NTEMAN system can locate the worker, identify objects around him, and communicate the information back to remote site. The information informs the subsequent interaction between experts, field workers, and computer networks in the field.

Non-System Support:

```
0. Fix computer network problem in the field
      1. Go to the computer network room (User task)
      2. Follow the manual to check each point step by step (User task)
      3. Correct the fault (User task)

Plan 0: do 1 - 2 in that order
        if error found
        then 3
```

System Support:

```
0. Fix computer network problem in the field (Equipped with wearable or
mobile computing device with sensors)
      1. Go to the computer network room (User task)
      2. Correct the fault (User task)
```

```
Plan 0: do 1
        if error found
        do 2
```

From the HTA, we see that task 2 in non-system support section is transferred to the remote expert team. The field worker simply focuses on following the instructions of fixing errors from the remote expert team and need not to traverse between checking the status of equipment and user manuals. The system exploits the information about the field worker's location and surrounding equipment to provide relevant information for the collaboration between field workers and remote network experts. Field workers can focus on doing the tasks with help from the remote expert without spending additional time checking manuals and so on.

2.4 SUMMARY

In this chapter, we have presented a review of related task analysis techniques and a discussion on the reason why we chose HTA to help us to identify tasks in existing context-sensitive mobile computing systems. We have also presented a review of existing context-sensitive mobile computing applications. These applications are categorised as office utilities, guides, social utilities, games, and field work. There are a number of important issues in these example application scenarios. Existing research in this area aims to simplify users' tasks using context information. The HTA showed in each application domain, that the number of user task can be reduced with context-sensitive mobile computing system support. Some user tasks in the system support scenario in each application domain are considered as implicit input to the system. The context information for an application can be considered as a subset of inputs that are used to perform the tasks that the application performs to help its user achieve a certain goal in a situation, namely, the implicit input that is not intentionally performed by the user to operate a system but conceived as input by the system (Schmidt, 2000). For instance, a user carrying a handheld computer is interested in the artefact in front of her in a museum resulting in the information about the artefact displayed on the computer. The user performs the task, "walk to the artefact interests her", is regarded as an input to the system to perform system tasks to suit the user's current situation. This is the definition of context that we use in this thesis.

These systems know what some user tasks mean! These applications utilize the sensed context information by their underlying system infrastructure to simplify their users' tasks. Users are required to wear specialist transmitters or carry mobile computing devices with them as personal identifiers while on the move. Also, we have found the context information exploited in context-sensitive mobile computing can be categorized as user's location, user's preferences, and surrounding resources (i.e. workstations, printers, people, etc.). We want to emphasis that location information about the user is the most widely used index into context in mobile computing applications so far. This reflects the importance of the relation between location, mobile computing devices, and users. The devices carried by the users have some means to sense their environment and make assumptions about user tasks by

utilizing the location information (Dix et al., 2000, Tuulari, 2000). From this previous work, we can make a number of observations: Firstly, we must understand and obtain the pattern of user's activities when performing specific tasks to achieve a certain goal under specific circumstances. Secondly, mobile computing devices and the underlying system infrastructure should be able to sense their user and environment. Thirdly, applications should act along the sensed context information and provide their users with task-relevant information. However, these scenarios implicitly tell us the trade-off between efficiency with system support and human factor issues. For instance, systems with active tracking sensor mechanisms make users concentrate on performing relevant tasks to deal with their current situations without the disturbance from the sensor operating tasks. However, users may lose control of their privacy. We also learned a lot about HTA. It does not capture social issues very well. For example, in the case of smart mobile phone interaction, the real issue is not work saved for users but annoyance to their colleagues. In the context-sensitive mobile computing game scenario, the HTA cannot address the issue of enjoyment.

In the next chapter, we describe the supporting infrastructures of each context-sensitive mobile computing system mentioned previously. In addition, the usability and social concerns that arise from each underlying infrastructure is discussed.

CHAPTER 3
PREVIOUS CONTEXT-SENSITIVE MOBILE COMPUTING SUPPORTING INFRATRUCTURES

In Chapter 2, we discussed previous work in the field of context-sensitive mobile computing. Our preliminary conclusion about the function of a context-sensitive mobile computing system is: Firstly, the system must know the meaning of user task performed in a particular situation. Secondly, the system must be able to sense the users and their environment. Thirdly, the system should transfer some user tasks to system tasks based upon the knowledge of user's activity pattern and sensing ability. A number of authors argue that there is more to context than location and immediate environment information (Brown, 1998, Schmidt et al., 1998). However, in this chapter, we focus on previous approaches that sense users' location and surrounding resources. In particular, we examine the context sensing technologies that have been utilized to support the applications discussed in the Chapter 2. The analysis focuses on how previous approaches obtain information about users' locations to identify surrounding equipment and services using sensors, mobile computing devices, and wireless communication. The survey is necessarily partial give the pace of development in this area. In addition, a number of researchers have argued that the lack of software infrastructure support causes the development of context-sensitive mobile computing systems a non-trivial task (Chen and Kotz, 2002, Dey, 2000, Hong and Landay, 2001, Kindberg and Barton, 2001). A need to tackle this issue has been brewed and several general software architectures have been proposed. We, therefore, review previous approaches in supporting software architecture for context-sensitive systems in an attempt to realise the challenges and common agreements in this field.

3.1 APPROACHS TO SENSE THE CONTEXT

In the previous chapter, we pointed out that previous context-sensitive mobile computing approaches can sense their user and environment. How could we make mobile computing systems more context-sensitive? Sensor development arguably offers an answer to this question. A sensor can be considered as a black box that receives input and provides output. The output is partially determined by the input. A sensor measures the world that is external to a sensor itself and transforms the reading into suitable information for users or computing devices. For instance, a thermometer measures the temperature of physical objects or the environment and transforms the phenomenon to numerical, colorific format, etc. A Global Positioning System (GPS) receiver can determine its location by measuring the signals from nearby satellites. A human is able to sense the temperature and transform the phenomenon to the feeling of hot, warm, cool, and cold. Also, humans can identify their own location by eyes seeing road signs or capturing the environment as an image and looking up their previous experience in memory (Dix et al., 1998). More generally, sensors can be divided into, "natural sensor" and "artificial sensors". In the physical realm, we use natural as well as artificial sensors to detect changes in our

surroundings. Mobile computing devices only use artificial sensors. From the context-sensitive mobile computing designer's point of view, it is important to divide sensor types into nature and artificial sensors. Designers must know how to transfer user tasks to system tasks to simplify their user scenario when designing a context-sensitive mobile computing system. To transfer user tasks to system tasks, a context-sensitive mobile computing system must sense the context in a certain situation. As an example, in the in/out board mentioned in Chapter 2, the tasks of finding a user's name and moving the indicator on the board to reflect the current in/out status in the office require that humans use a natural sensor, their eyes, to locate his/her name on the board and then move the mark to indicate the in/out status. In this case, systems can transfer the user tasks of finding their name and changing the status on the board using an iButton and an artificial sensor. Using the iButton, the system knows who the user is and changes the in/out status for the user. Figure 3.1 illustrates our view of interaction between a user, a mobile device, and sensors.

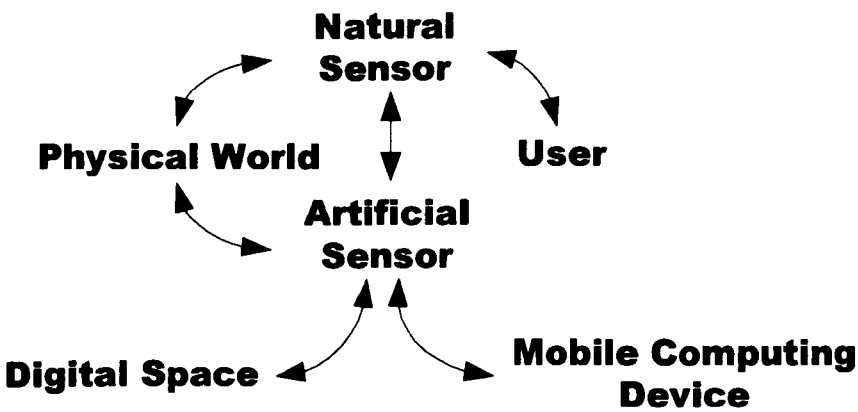


Figure 3.1 Interactions Between a User, Mobile Computing Device, and Sensors

The following sections describe the techniques that most existing context-sensitive mobile computing systems exploit to sense their user's location and surroundings. Previous work that implemented these sensing techniques is described in the section 3.2.

3.1.1 LOCATION-SENSING TECHNIQUES

In this section, we briefly describe existing location-sensing approaches in context-sensitive mobile computing. Using these techniques, mobile computing devices can sense their users' locations and provide location-dependent information and services to the users. User tasks such as figuring out the current location, directions to next destinations, etc. can be passed to systems that can recognise the user's current location. We categorized the techniques into direct identification, cellular proximity sensing, and differential positioning.

3.1.1.1 DIRECT IDENTIFICATION

This relies upon the system directly identifying the presence of the user. There are currently several international projects looking at the use of image analysis to support this. Others exploit electromagnetic techniques to provide location information for context-sensitive applications. These systems resemble the applications that are used to protect the books in libraries and the merchandise in shops (Raab et al., 1979). It is possible to identify a magnetic signature as it passes through a pair of detectors. These can be placed at strategic locations to monitor the user's movements. For example, they can be placed at doors to detect the entrance or the exit of any item which carries the signature. This approach has a number of benefits. In particular, the stripes that carry the signature can be very unobtrusive. However, the detectors are relatively expensive and must be carefully positioned to provide the coverage that is required to trace a users' changing location. There are further problems. For example, electromagnetic methods suffer interference from computer monitors and metal structures (Harter et al., 1999). The ethical problems that are raised by this technology have recently been avoided through the use of tokens, such as iButtons, and their associated readers. Users explicitly indicate their location by depositing the token in an appropriate reader each time that they change location. They, therefore, can choose when to disclose their location. This raises an obvious usability issue. The drawback of this approach is that users frequently forget to explicitly indicate their movements (Salber et al., 1999).

3.1.1.2 CELLULAR PROXIMITY SENSING

Cellular Proximity Sensing technique relies upon fixed transmitters that generate a unique identifiable signal, which has a precisely determined range. The concept of this approach is that if a mobile computing device detects this signal then it must be within the area, or cell, that is covered by the transmitter associated with that signal. For instance, this technique can be used to identify the gross location of a device within a Group Spatial Mobile (GSM) network. The GSM infrastructure supports most of Europe's mobile telephones[4]. Similarly, wireless local area network (WLAN) access points can be used to provide a location proximity sensing service. Mobile computing devices that are able to access information over these access points are guaranteed with this sensing service if an appropriate identification mechanism (i.e. mobile computing device or access point identity) is implanted.

3.1.1.3 DIFFERENTIAL POSITIONING

In contrast, differential Positioning exploits a number of fixed transmitters each generating a unique signal. If the device can detect those signals then it is possible to use a range of triangulation algorithms to calculate its position relative to the sources of those signals. This technique is at the

[4] http://www.gsmworld.com/news/statistics/index.shtml GSM official website.

heart of the Global Positioning System (GPS) (Dana., 2000). There are two outstanding subcategories of triangulation technique, namely, lateration and angulation.

Lateration techniques rely upon measuring the distances from an object that a system wants to locate to at least three signal transmitters to calculate the object's location in two-dimension space. These signal transmitters are located at non-collinear positions. In the case of locating an object in three-dimension space format, four signal transmitters are required. The distance between an object to a signal transmitter can be derived from direct physical measurement, for example, using "string rulers". Two more effective distance measurement methods rely upon signal time-of-flight and attenuation. The signal time-of-flight method is based on the known velocity of signals (i.e. radio frequency or sound). Distance information can be obtained by measuring signal travelling time. Similarly, the signal attenuation method is based on the idea that signal strength decreases as the distance from the transmitter increases (Hightower and Borriello, 2001b). The ratio of signal strength to distance is fixed to $1/r^2$ (r: distance from signal source to the measuring object). In this case, the distance between an object and a signal transmitter can be obtained by measuring the arriving signal strength at the object. Figure 3.2 illustrates the concept of locating an object in both two and three dimension using lateration approach.

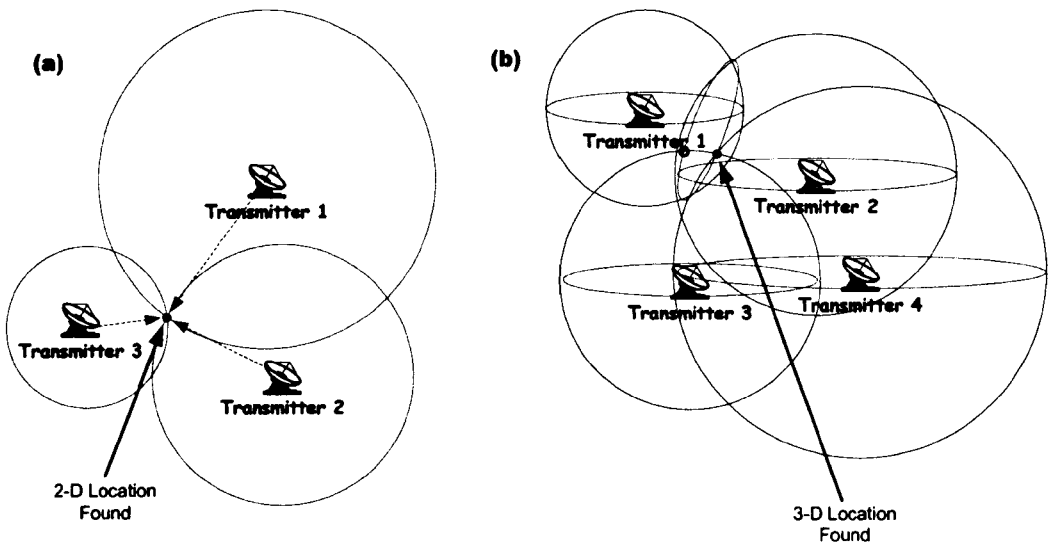


Figure 3.2 (a) 2-D Lateration Location Sensing (b) 3-D Lateration Location Sensing.

The Angulation technique is based on measuring the angle of a signal arriving at two receivers each equipped with an array of phase antennas. The distance between the receivers must also be known to calculate the location of an object in two-dimension space (Hightower and Borriello, 2001a). Once the angles are obtained, the location of an object can be derived from triangulation calculation. Figure 3.3 shows the concept of angulation technique. However, these techniques focus on and measure the direct signals transmitted from the signal transmitters instead of reflected ones. Signal reflection in the environment causes difficulties in measuring distances between an object and signal transmitters using time-of-fly, angulation, and attenuation techniques.

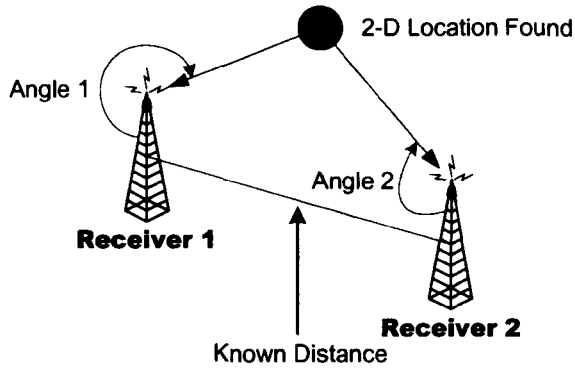


Figure 3.3 Angulation Location Sensing

3.1.1.4 SIGNAL STRENGTH ANALYSIS

This approach relies upon deriving the most likely location from the signal quality data set that records electromagnetic characteristics, for instance, the signal-to-noise ratio (SNR) between the client mobile computing device and remote base signal transmitters (i.e. access point), while the user is carrying and moving around in the building. In this case, each location is associated with a set of SNR values. This location-SNR data set can be used to build up a lookup table. The mobile computing device's location can be computed by matching the received SNR values and the corresponding location information in the table. The claimed effectiveness in discriminating between locations is roughly 10 feet (Castro et al., 2001). However, the performance is highly depending on the building layout and number of base signal transmitters and by changes in the infrastructure, for example, electromagnet signal interferences when objects are moved.

3.1.2 ENVIRONMENT-SENSING TECHNIQUES

As we noted in Chapter 2, context-sensitive mobile computing systems should be able to sense their user's surroundings such as computing resources, other people, etc. To sense other people in the user's vicinity, the systems not only can sense the user's location but also establish a location data sharing mechanism to notify other appropriate parties about who the nearby users are (i.e. Active Badge). Sensing surrounding resources, a context-sensitive mobile computing system need to perform tasks as a human looking around the immediate environment and deciding which resources can be used to help finish the current tasks. To do this, a context-sensitive mobile computing system needs to recognise what resources are nearby. Most of the existing approaches are utilizing electronic tags such as iButton, RFID, infrared beacon (i.e. Active Badge, ParcTab, cooltown (discuss more in section 3.2.3) URL tag, etc), ultrasound (i.e. Active Bat), etc (Kindberg et al., 2000). These tagging techniques can be categorized into cellular proximity sensing and direct identification as descried in the location-sensing section. RFID and infrared beacon have short range wireless communication range and

therefore should go into cellular proximity sensing. As for iButton, it needs its user physically docks the button on the reader and should go into direct identification.

In brief, to sense the environment, context-sensitive mobile computing systems rely upon the sensor infrastructures that connect physical objects with digital representations or services. There are currently two means of achieving this. Firstly, electronic tags or transmitters that are able to transmit unique signals are used to bridge the physical and digital space. The tags can be placed or embedded within physical objects and environments. The signals that are transmitted from the tags can contain information about the correlation with digital representations or services. Secondly, in contrast to the self-contained sensor signal approach, most of the existing systems in this area exploit tags that emit signals. These need to be translated to obtain the correlation with digital representations or services. The tag's correspondent receiver obtains signals from the tags and passes the signal values to a translator module that looks up to get the correlation with digital meanings.

3.2 EXISTING CONTEXT-SENSITIVE MOBILE COMPUTING INFRASTRUCTURES

This section reviews a number of different engineering approaches that have supported the development of context-sensitive mobile computing applications mentioned in the previous chapter. We examine each project, in particular, their system architecture, context information that they can deal with, system implementation, and usability issues.

3.2.1 ACTIVE BADGE

Arguably the best known context-sensitive mobile computing system is the Active Badge system (Want et al., 1992). This depends on a network of infrared "base sensors". Base sensors are set at different fixed positions around the host building. People are located by wearing a badge that periodically (~ 15 seconds) transmits an infrared signal containing a unique identifier. Equipment can also be located by attaching a badge that has longer signal transmitting interval (~ 5 minutes) than for people. This is due to the assumption that equipment is less "mobile" than people (Harter and Hopper, 1994). These periodic infrared signals are detected and received by base sensors. The base sensors are connected to a wired local area network. A master station (workstation) is designed to poll each base sensor for badge "sighting" events. Once the infrared signals are received and identified by base sensors, a sighting event is then sent to the server, which runs on the master station. The badge's location can be obtained by matching the physical locations of detected sensors at the server. The server has four communication layers, namely, network control, representation, data processing, and display interface. The network control layer is designed to poll the base sensors installed throughout the building. The representation layer is designed to format valid data extracted from the base sensor network into a triple (badge ID, location, time) data structure. The data processing layer is responsible

for minimizing network traffic by only reflecting the changes in the gathered badge activities. The display interface layer takes the previous three layers as input and provides a display function either in textual or graphical form.

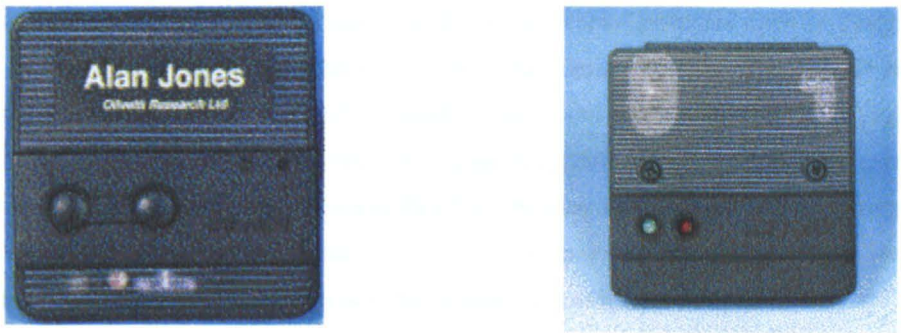


Figure 3.4 Active Badge (Left) and Base Sensor (Right)
(Source: http://www.uk.research.att.com/ab.html)

In the final system, these badges had limited processing capabilities. They could also receive messages from the network of base stations. One of the limitations of diffuse infrared signals is that they can potentially be received at many different positions within a room. The Active Badge system was, therefore, extended to a hybrid approach exploiting radio signals. Short-range transmissions are used so that the wearer's position can be determined more precisely. This is then encoded and relayed back via an infrared transmission to the base station. However, this approach requires considerable initial investment to establish the necessary network of base stations that receive signals from each person's badge. Each user is also required to wear a badge in a prominent position so that is can receive and transmit the necessary signals. Some authors argued that there are social and cultural issues associated with the involuntary location information disclosure (Want et al., 1992). The users react by adopting a number of informal consensuses to determine when and where an Active Badge would need to be worn.

3.2.2 PARCTAB

The ParcTab system has many similarities with the Active Badge approach (Schilit, 1995). Again, infrared communications implement cellular proximity architecture. The infrared signal in both Active Badge and ParcTab system can be reflected and contained by the partitions in a room and therefore each room can be treated as a cell. Users carry specialist devices that resemble pagers. As with the Active Badge system, these devices communicate with fixed infrared transceivers that are connected to conventional wired networks. The location sensing functionality of these devices is slightly more limited than the Active Badges. They do not support short-range radio signals (Want et al., 1995). However, it is important to emphasize that ParcTabs more closely resemble PDAs than Active Badges. This can be seen from Figure 3.5.

Positioned over the display is a touch sensitive panel. The tab also includes three finger-operated mechanical buttons that can be used individually or in chords. The unit includes a piezo-electric speaker that permits a number of different tones to be generated by applications. The ParcTab simply acts as an information terminal with a graphical user interface. The applications are hosted and run on the remote server. This architecture resembles the Model-View-Controller (MVC) design pattern proposed by Buschmann et al (Buschmann, 1996). This design pattern separates an interactive application into model, view, and controller modules. The model module holds the core function and attributes of an application. The information is displayed by the view module and the user input is dealt with by controller module. In the case of ParcTab, the user interface on the device comprises the view and controller modules whereas the model module is implemented on the remote server. This approach has influenced our design approach. More detail is provided in Chapter 5.

The ParcTab server implementation contains an infrared gateway, tab agent, and shell. The infrared gateway manages the infrared signals sent and received between the infrared transceivers and the ParcTab agents. The ParcTabs can send infrared encoded commands to the infrared transceivers. The commands are forwarded to the infrared gateway and then delivered to appropriate ParcTab agents, once the infrared transceivers receive it. Conversely, the ParcTab agents send responses to the infrared gateway when the application executions finish. The responses are encoded into infrared data packets and then sent to infrared transceivers through appropriate serial port that connect them. The infrared transceivers then broadcast the data to the ParcTabs within the communication cell.

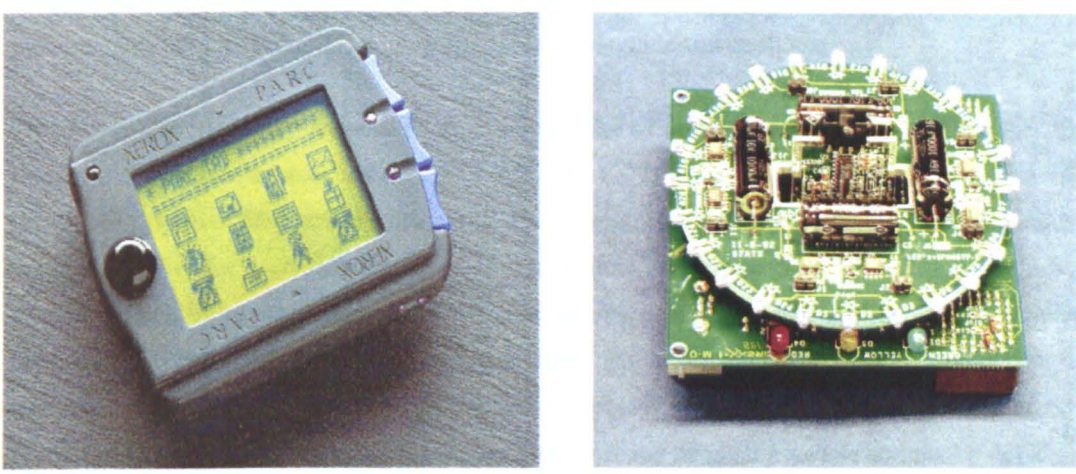


Figure 3.5 ParcTab (Left) and Infrared Transceiver (Right) (Source: http://www.ubiq.com/parctab)

Each ParcTab agent process represents a ParcTab and tracks its location. When a ParcTab moves to a new cell, it receives an infrared identifier from the infrared transceiver located in the cell and its correspondent agent updates the location information hosted at the location service server in order to notify the applications running on remote hosts to provide location-dependent information such as surrounding resources and people.

The shell provides a user interface for operating other tab applications on a ParcTab as well as an interface between tab agents and the tab application. A ParcTab agent can initiate a shell to provide a graphical user interface on the user's ParcTab and waits for the user to select an application. When an application is selected the shell can register the application with the corresponding ParcTab agent though the `AppControl`, which is an interface between a ParcTab application and a ParcTab agent. The interface provides four functions: register, suspend, resume, and quit. When an application calls "suspend" or "quit" command, the agent switches the control back to the shell. When "resume" is selected, a suspended application can be started from the previous state. If an application runs into system deadlocks, the agent escape event can be sent to force the agent to suspend the current application and switch back to the shell.

ParcTab system was a predecessor for many proposed interactions in context-sensitive mobile computing but also demonstrated a working communication architecture for mobile computing devices, sensors, and wireless communication. Unlike the Active Badge system, the passive location sensing approach provides the ParcTab user with full control of their location disclosure.

3.2.3 INFRARED TAGGING

This approach exploits various infrared transmitters, which can transmit unique identities, as indices of locations and objects in the physical environment. Mobile computing devices can sense these infrared beacons and use them as links to digital information to present to their users. We look at two pioneering developments of this approach, indoor Cyberguide and cooltown.

The indoor Cyberguide system makes more limited use of infrared communications than the Active Badge and ParcTab approaches (Gregory Abowd, 1997, Long et al., 1996). Rather than use a networked system of base sensors, this application simply used infrared transmitters to communicate a unique identifier for each location. Infrared transmitters are mounted on the ceiling of rooms and corridors. Each transmitter sends out a unique code repeatedly as a location identity. A number of hand-held and palm size mobile computing devices, Apple Newtons and Palmtop computers, were equipped with a specialist unit consisting of a separate infrared sensor and a Motorola 68332 processor. Once the users walk into the infrared transmission range, the mobile computing devices can decode infrared signals into location identities and update the location on the map on the devices' screen. This architecture then enabled people to trace their position within a number of cells made up from the transmitters. In their original implementation, each mobile computing device caches all of the location dependent information. These infrared beacons are utilized to index into the stored data. Similar to ParcTab, the indoor Cyberguide exploits passive location sensing. However, there is no additional network communication required to provide location information since the users' handheld device are made aware of all the identifiers for each of the transmitters before being used. This system simply relies upon the user's equipment detecting signals that are generated from infrared beacons in their environment. This is an important distinction. These infrared beacons can be low cost signal

transmitters, such as domestic remote controllers, rather than the active sensors of pioneering systems such as the previous mentioned Active Badge system. A number of further problems restrict the application of the indoor Cyberguide approach. The use of custom made infrared units, rather than the on-board infrared ports of mobile computing devices, increases the costs of the system. The absence of independent wireless communications together with the passive nature of the fixed transmitters also prevented other users from obtaining any information about their colleagues. This imposes considerable limitations on the users' view of their context.

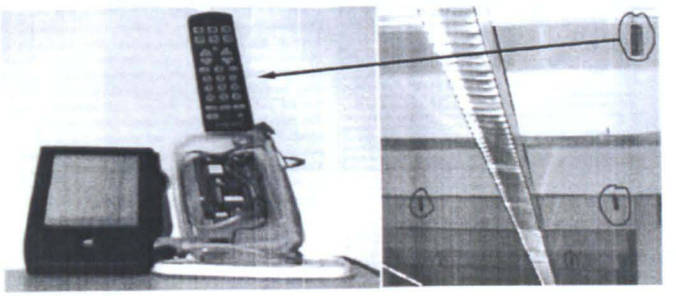


Figure 3.6 Indoor Cyberguide Implementation (Source: (Long et al., 1996))

Hewlett Packard's cooltown project is based upon the idea that every user, ever object and every location in physical space will have a web-based representation (Caswell and Debaty, 2000, Kindberg et al., 2000). This was motivated by the advance of web technology, wireless networks and portable devices. As an example, the visitors to cooltown Museum will be able to access information about the exhibits from their mobile computing devices. These devices can automatically download this information from the web using a URL that is associated with an infrared beacon placed close to each exhibit. Visitors implicitly select the web reference by moving within the transmission range of an infrared beacon. In addition, the cooltown project is augmented with web-enabled devices, such as printers and projectors. Users can exploit their mobile computing devices to print documents by sending URLs to an appropriate printer. Similarly, documents, for example, presentation slices, can be sent to a web-enabled projector in a meeting. This project exploits the same passive infrared sensing as the indoor Cyberguide. It also builds specialist infrared transmitters that can emit predefined URLs. Rather than focusing on low level wireless communication and mobile computing device technology, this project concentrates on extending the existing web server architecture to suit their revolutionary view of bridging web technology and physical entities. The cooltown development team built a toolkit, Web Presence Manager (WPM), which provides designers with support for the creation, management and hosting of cooltown web-presences of people, places, and things. The web-presence manages information and services about a physical entity, such as a person, a place, or a thing. Each web-presence can interact through standard web protocols. As an example, companies can have a web-presence that manages the web-presence of their employees, their places (i.e. buildings, offices, etc), and their things (i.e. printers, projectors, etc). A number of different deployment models are possible along this concept (network, 2002).

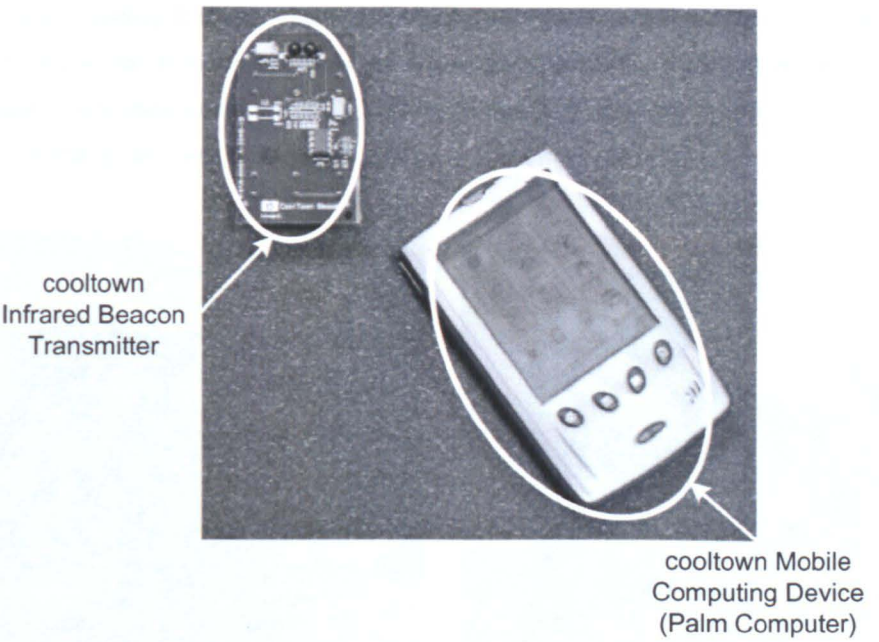


Figure 3.7 cooltown System Components

(Source: (Kindberg and Barton, 2001) & http://cooltown.hp.com/beacon_full.htm)

The cooltown project is a pioneering vision that looks beyond what might be feasible in the short to medium term. However, there are some limitations in their approach. It has not yet considered the engineering of an appropriate network topology to support the more visionary applications. What happens when the users are out of wireless network cells? The implementation details are not sufficient for designers in this area to implement a working system without recourse to the specialist hardware of previous approaches.

3.2.4 RADIO FREQUENCY IDENTIFICATION (RFID) TAGGING

Want et al in Xerox PARC proposed using electronic tags to bridge physical and digital space (Want et al., 1999, Want and Russell, 2000). This project is an extension of the previously mentioned ParcTab. Electronic tags are attached to physical objects in the same manner as infrared tagging. The users carry their mobile computing devices equipped with a radio frequency identity (RFID) reader and radio wireless communication interface to detect and get relevant information about objects from a remote server. The infrared transceivers are strategically placed within the environment for location detection in order to obtain more information about the tagged objects. There are a number of advantages in this approach. RFID tags do not require extra power to keep functioning. A tag reader can energise the tag by inductive coupling between the coil on the reader and the tag (Want et al., 1999). They are small and robust. These tags can be installed invisibly on physical objects without affecting tag detection. There are, however, some restrictions in this approach. For instance, the detection range of RFID tags is short (up to 10cm). As mentioned previously, the tags can be added to

physical objects invisibly. The users may have to keep their mobile device very close to the physical objects and "guess" the position of the tags. This might fail the tag detection because there is no explicit visual sign to indicate the exact tag position. Extra RFID tag readers are required to install on the mobile computing devices.

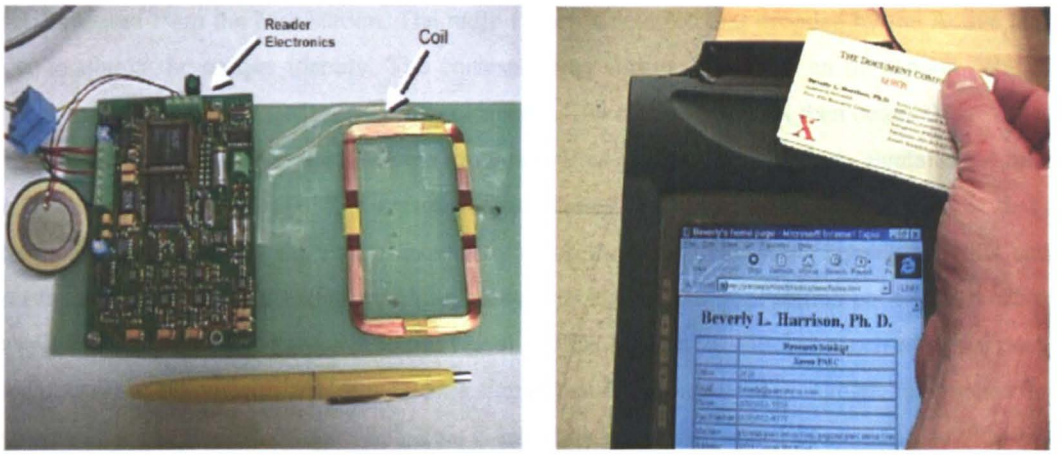


Figure 3.8 ParcTag Reader (Left) and ParcTag Mobile Computing Device (Right)
(Source: (Want et al., 1999))

3.2.5 ACTIVE BAT

Most indoor context-sensitive mobile computing systems rely upon infrared transmissions to implement the cellular architecture that was introduced in the previous section. However, the Active Bat system is an exception (Harter and Hopper, 1994). This approach relies upon the user carrying a device that contains a radio receiver, an ultrasonic transducer and associated controllers, to locate the user. The Active Bat system can also trace equipment by attaching Active Bats to them in a manner similar to the Active Badge system. Figure 3.9 shows the Active Bat system devices.

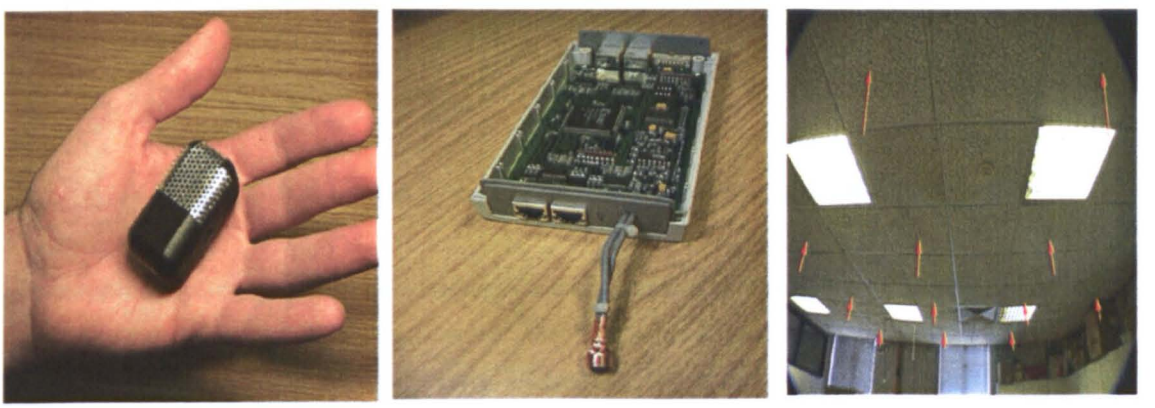


Figure 3.9 Active Bat (Left), Ultrasound Receiver (Middle), and Ultrasound Receiver Mounted in the Ceiling (Right)(Source: (Ward et al., 1997))

This system also relies upon a number of ultrasound receiver units. These are placed on the ceilings of the rooms that are to be instrumented. Each receiver is connected to form a daisy chain of similar receivers. The Active Bat system also consists of a base station. This periodically transmits a radio signal that contains a unique 16-bit identifier and a reset command to each receiver to reset the clock on it through a wired network. The receivers monitor the ultrasound for 20ms once they receive the reset command from the base station. The radio signal is detected and decoded by the Active bats in order to obtain the unique identity. The corresponding Active Bat emits an ultrasound "chirp" in response to this message by triggering its transducers. The closest receivers then detect these "chirps". When the ultrasound arrives, the receivers' on-board-circuit stops the clock and calculates the time of the first received signal peak. The reason for calculating the first received signal peak is that the reflection of ultrasound in the environment may reach the receiver as well. Signal reflection causes error because it must travel a longer path than a straight signal. The base station polls each receiver to retrieve the time interval between the reset signal and the moment of the first received signal peak. If three or more non-collinear receivers detect a signal then it is possible to triangulate the precise three-dimensional location of the Active Bat. This is based on the known speed of sound in air. Therefore, the distance from the transmitter to each receiver can be calculated. Figure 3.10 shows the Active Bat architecture.

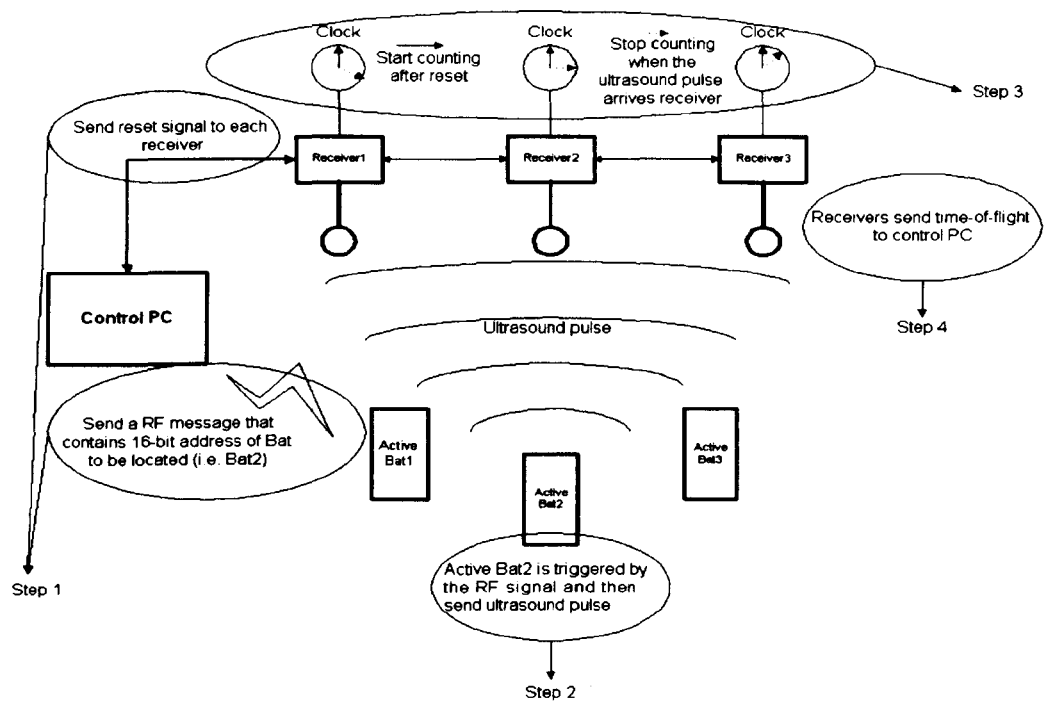


Figure 3.10 Active Bat System Architecture. (Source: (Cambridge, 1998a))

This approach offers considerable advantages over the more widespread use of infrared signals. It is more accurate (Harter et al., 1999). However, there are some drawbacks. In spite of the small size of the Active Bat, each user must still carry a dedicated object that conveys positional information. Perhaps more seriously, there is a considerable overhead is wiring each building to accommodate both the base station and the array of ultrasound receivers.

3.2.6 RADIO FREQUENCY LOCAL AREA NETWORK (LAN) SENSING

Radio signals offer a further alternative to either ultrasound or infrared. This approach currently relies upon a cellular architecture (Davies et al., 2001). For example, Agere Systems' WLAN[5] is a commercial mobile radio network. Each mobile computing device communicates with a base station using a PC card that currently supports an effective bandwidth of 11Mbps using the IEEE 802.11b[6] standard. The base station typically has a range of approximately 100-200 meters depending on the construction of the building. It is, therefore, possible to gain a very rough approximation of the users' position as they move between these relatively large cells. It is also possible to triangulate a users' position using the broadband radio signals (Bahl and Padmanabhan, 2000, Castro et al., 2001). This offers considerable advantages. The mobile network would connect the user to remote desktop resources and locates them within their environment. Agere represents one use of radio frequency technology to support context-sensitive. The Bluetooth[7] project represents another. Bluetooth is the codename for a short-range radio frequency system that is being developed to link mobile computing devices, such as mobile phones and other portable devices. Bluetooth connections are designed to work over distances of up to 2 or 3 meters rather than the 100-200 meters supported by Agere. From this it is readily apparent that it offers considerable potential for more fine-grained cellular architectures than those supported by the Agere architecture. Bluetooth is intended to be a standard for communication between mobile computing devices. It also has the benefit that it is likely to become a feature of many different commercial products.

Microsoft is working on RADAR, which is a triangulation technique using radio LAN. This approach is based on the concept that the received signal is strongest when the user is near the base station and the weakest signal is received when the user is distant from the base station. Before the RADAR system can track users, it must record information about the RF signal strength (SS). It can be regarded as a clue to the user's location as we described in section 3.1.1.4. The SS can be extracted from Agere firmware. The procedure for collecting data is as follows: Firstly, synchronize the clock on the target mobile computer and base stations. Secondly, the mobile computer starts broadcasting UDP packets. Each UDP packets has 6-bytes content and is equally spaced between packets. Thirdly, once the base stations receive the packets from the mobile computer, the SS data and timestamp are recorded in the format of `(timestamp, base station, signal strength)`. Fourthly, the user, who carries the mobile computer, clicks on a map of the floor to show the current location and then the user's coordinates `(x,y)`, facing direction, and timestamp are recorded in the format of `(timestamp, x, y, direction)`. The information is used to build the search space. In data processing stage, a program obtains the mobile computer's real-time layout information of the floor

---

[5] http://www.agere.com formerly known as Lucent's WaveLAN.

[6] http://grouper.ieee.org/groups/802/11/ IEEE 802.11 official website.

[7] http://www.bluetooth.com Bluetooth official website

and determines the exact and closest matches and then the mobile computer's location can then be "guessed".

Figure 3.11 RADAR System (Source:(Research, 2000))

Unfortunately, this remains a subject of active research. The multi-path transmission problems that affect GPS and ultrasound signals cannot easily be resolved for broadband radio transmissions. Differential signal detection is subject to distortion from the movement of objects in the environment. There are also problems of interference between 802.11 series and Bluetooth standard (Punnoose et al., 2001). In anticipation of the results of this research, the resolution offered by broadband radio signals is too large to support most context-sensitive mobile computing applications particularly in the indoor environment.

3.2.7 DIRECT IDENTIFICATION (iButton)

The final approach to providing location information is, in many ways, the most obvious. Rather than using radio signals or infrared transmissions, it is possible to use image analysis techniques to identify individuals as they move around a building. Security cameras can be upgraded to give significant coverage without the overheads of rewiring an entire building. However, the technological and ethical problems with this approach are considerable. In all of the other techniques, it is entirely possible for users to "hide" their location for some or all of their activities. It is not clear how users might "opt out" from direct visual monitoring. An alternative approach is for the user to explicitly provide the system with location information. For instance, Georgia Tech's Context toolkit exploits iButton technology (Dey, 2000, Salber et al., 1999). These steel-plated microchips provide 64k+ memory and a form of the Java virtual machine. They can be programmed to encode information about their owners. This can be read when the user inserts the button into a dedicated reader. Placing readers at various locations within a building provides location information. The relatively small size and weight

of these "wearable" devices makes them an attractive alternative to signal detection. However, initial studies have revealed the problems of relying on the location information that can quickly become obsolescent as users "forget" to update their position.

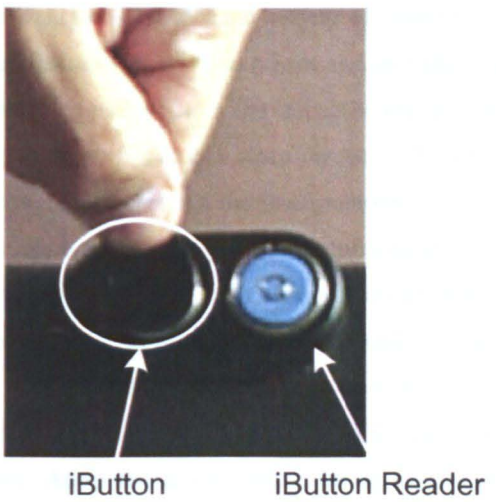


Figure 3.12 iButton (Source: http://www.cc.gatech.edu/fce/contexttoolkit/)

3.3 SOFTWARE ARCHITECTURAL SUPPORT FOR CONTEXT-SENSITIVE SYSTEMS

In the previous section, we described existing context-sensitive mobile computing systems in terms of their hardware architectures and the use of mobile devices, wireless communication, and sensor technologies to capture context data and disseminate tailored information based on the context. The current consequence is that this type of system has been developed in an ad hoc manner. The development of context-sensitive systems is heavily influenced by the underlying architectures. A number of researchers have argued that the lack of software infrastructure support causes the development of context-sensitive mobile computing systems a non-trivial task (Dey, 2000). A need to tackle this issue has been identified and several general software architectures have been proposed. We, therefore, review previous approaches in supporting software architecture for context-sensitive systems in an attempt to realise the challenges and common agreements in this field.

3.3.1 Context Toolkit

The Context Toolkit was developed to support the acquisition and delivery of context information when building context-sensitive applications (Dey, 2000). The aim is to make context-sensitive applications easy to build. It provides four architectural building blocks to simplify the task of

implementing context-sensitive applications. These are context widgets, context interpreters, context aggregators, and context-aware services.

A context widget is in charge of acquiring a specific type of context information and makes it available to applications. Similar to Graphic User Interface (GUI) components, a context widget is designed to be reusable and changeable to suit the variety of context sensing. It can encapsulate the details about how to access a particular sensor that senses and provides relevant context information to context-sensitive applications. This also reduces the impact when the underlying sensor technologies are changed. In addition, a context widget can store the sensed data and make it available to the applications. A context interpreter implements the interpretation of context information from one or more context widgets. An example is converting from an infrared signal to a physical location. Other example is that combining different context information from all the context widget, for instance, room schedule, people's presence, etc, in a conference room can determine that a meeting is carrying out. A context aggregator is designed to take all the context information for an entity. It is a combination of more than one context widgets. A context aggregator acts as a proxy to context for context-sensitive applications. Applications can subscribe or poll context aggregators in order to trigger corresponding context-sensitive behaviours. The context-aware services building block is to implement context-aware services that can be shared by multiple applications such as displaying a message, for example. This reduces the burden of implementing similar services for each different application.

The Context Toolkit not only provides its users the building blocks to construct context-sensitive applications but also gives them a high level view of this type of application. The users of the Context Toolkit can also tailor or add building blocks to the architecture to suit their needs. The underlying concept is that the existing components provided by the Context Toolkit may not be able to address all the requirements when developing context-sensitive applications. For example, it cannot provide all of the components that sense and interpret a variety of contexts.

3.3.2 Context Fabric

The Context Fabric is designed to address the problems of how to model, store, and distribute context information. It also proposes a context specification language to help developers describe context queries at an abstract level. Privacy issues are especially concerned in the Context Fabric. The motivation of this approach is to address the social concerns when users use this type of system (Hong and Landay, 2001).

There are two data model of the context data store module in the Context Fabric: logical and physical context data model. Context information is represented using entities, attributes, relationships, and aggregates based on the logical data model. Similar to the abstract representation of physical world proposed by cooltown, the entities in the model are simply people, places, and things. Properties of the entities, such as the name of a person, are described using attributes. Relationships describe the setup

between entities. For example, a place could contain things and people. People can be in a specific place. The aggregate in the Context Fabric is similar to the aggregator in the Context Toolkit. It groups entities and provides a more complex context representation. The physical data store handles the context data storage. The Context Fabric emphasises the privacy control in the distribution of context data. For example, personal context information can be stored in a user's personal device, such as handheld computers or mobile phones. The distribution and reuse of context data is rather similar to the Context Toolkit. This mechanism guarantees the separation of applications and their underlying sensors. Therefore, it can reduce the impact when the underlying sensors are changed. The context specification language follows the approach that SQL does for relational databases. Developers can author a query like, "What is the nearest printer to me?" It can be processed by the context services in the infrastructure. The context specification language can also help developers define events, such as "Notify me when it is raining outside."

The Context Fabric focuses more on the data modelling rather than the context acquisition in the Context Toolkit. It provides developers more detail about how to store the context information as well as how to retrieve it using the context specification language. Their emphasis on the privacy control issues is currently being implemented. Developers can assign restrictions on certain context data and queries.

3.3.3 Stick-e Notes

Stick-e Notes is based on the concept that application developers can attach entities to specific contexts. The predefined context-sensitive services will perform once the entity enters the attached context. This system provides a general mechanism for developers to decide which context information should be used and they can author rules to specify what actions to take when a particular combination of context is identified. For example, a developer can create a note, "when a user arrive the conference room on the first floor at 16:00 display the handout of the ongoing talk on her PDA". This system provides software architecture to help capture and process context information. There are three components in the system: triggering, execution, and sensor components (Pascoe, 1997). The triggering component gathers context information for an entity, for example, a person, and matches the context to the predefined stick-e notes. The relevant actions will be activated once the current context of the entity matches the note. The execution component can be existing programs. Developers can assign which existing program should be executed once the current context matches the note they defined. The sensor component is responsible for sensing the changes in physical or digital operating environment and passing them to the triggering component.

The aim of this work is to help non-programmers to build context-sensitive services easily. However, the lack of supporting sensing technologies and sensor data interpretation mechanism forces the supporting semantics for authoring rules to be limited to the programmers.

3.3.4 Cooltown coolbase

As we described in section 3.2.3, HP cooltown leverages off of the most recent hardware technologies in mobile computing and sensor development to form an environment for context-sensitive applications. It also exploits a web model for supporting the delivery and presentation of context information to applications. The research team implemented a platform, coolbase, to help developers build applications based on their cooltown environment. Similar to the Context Toolkit and Context Fabric, coolbase consists of a number of modules: coolbase appliance server, esquirt, web presence manager, beacons (Oppermann and Specht, 1999).

The coolbase appliance server hosts the services provided by web-enabled appliances. The appliances and their services are presented in web pages as interfaces to other appliances. This enables the appliances to interact and be invoked by other web-enabled devices. For example, a web-enabled printer can invoke tech support services such as a toner change request or process the printing job ordered by a user through a web-enabled PDA. The esquirt module defines an interaction model between web-enabled appliances. It enables mobile computing devices, such as PDA, or personal devices, such as a wristwatch, to act as a universal remote controller for the services provided by other web-enabled appliances hosted in the coolbase appliance server. For instance, a user carries a PDA and receives a URL for the website where the pictures that she took with her friends in a birthday party have just been posted. She may want to have the pictures printed out so she can simply beam the URL to a colour photo printer nearby using infrared or Bluetooth. In the meantime, the interface for the printer is shown on her PDA in a web page form. Therefore, she can choose the preferred format and send the request to have the picture printed out from the printer. Esquirt also implements the functions to receive and decode the signal from the cooltown beacon (see section 3.2.2). The web presence manager (WPM) implements the way to present views of an entity such as a person, a place, or a thing. Similar to the context aggregator in the Context Toolkit, the WPM gathers the context about an entity and presents the information in a web page. For instance, a user walks throughout the cooltown environment; she can see a web page on her PDA or nearby wall-mounted flat panel screen about the available services in the environment. The cooltown beacon is a hardware implementation of a short range wireless transmitter. It also implements a protocol to embed data such as URLs and device descriptions into the signal.

Unlike the Context Toolkit and Context Fabric approach, the cooltown coolbase emphasises how to display the current context of an entity and exploits URLs to encapsulate the sensed context data and relay it in a ubiquitous uniform way. It also demonstrates the new form of interaction between web-enabled appliances, simply said an invocation with reflected user interface. However, there is not enough information available about how the cooltown environment developers adapt new sensor technologies to obtain varieties of context.

3.3.5 Solar

We consider that Context Toolkit follows an operating system approach whereas the Context Fabric exploits the concept of a database system. Cooltown envisioned the future interaction model between web-enabled appliances and their users as well as the data communication as based on a web model. Although there are a number of similarities such as sensor data reusing, interpreting, and fusion in Solar, the architecture mainly focuses on the sensor data publishing and consuming (Chen and Kotz, 2001, Chen and Kotz, 2002). It is more like a messaging service. Solar considers sensors as information sources. The data about the changes or status in the digital and physical environment obtained from the sensors are treated as events. The information sources are implemented as objects. There is a component, operator, in Solar. It is an object that subscribes, process, and publishes event streams. The behaviour of an operator is similar to a context interpreter or aggregator in Context Toolkit and cooltown coolbase. The Solar proposed an operator graph to help developers to draw the data flow and transformation between objects. This provides them an abstract view of a context-sensitive system. There are three types of nodes in a Solar operator graph: sources, operators, and applications.

As mentioned, the sources are the sensors used in context-sensitive systems. The operators receive input events from the sources and publish the converted results to other operators or applications. The operators can be categorised as four different types: filter, transformer, merger, and aggregator. A filter can output results based on the rules defined by developers. For example, a filter can help to obtain a particular user's location instead of everyone's from a context-sensitive locating system. A transformer converts event from the sources to appropriate formats and make them available to other objects. A merger acts as a hub that a number of sources (sensors) are connected to. The advantage of this operator is that it gathers multiple sensor events so applications need not poll or subscribe each sensor every time when there are changes at the sensor. An aggregator simply outputs an arbitrary event based on the events in one or more input events. For instance, an aggregator can receive events from a merger or a source and only output events to the applications that subscribe the aggregator when there is change at the merger or the source. The applications in an operator graph are the end of the graph. They can subscribe to one or more event from operators and act upon the incoming events.

The Solar system provides developers a model to deal with the context information and relations between abstract event handlers. The advantage of this approach is to enable developers to plan their systems before really implement them. All the objects in an operator graph can be reused so the graph can grow to fit future developments. However, the limitation is that the lack of practical implementation helps for HCI developers to implement a functioning system.

3.4 SUMMARY

In this chapter, we have presented the existing location-sensing and environment-sensing technologies in most context-sensitive mobile computing systems. In addition, we have described the underlying infrastructure implementations of previous approaches and the usability issues when using them. We have found that a number of issues exacerbate the development of context-sensitive mobile computing applications. These can be summarized as follows:

- They have not been integrated with existing mobile computing technology and have relied upon additional hardware;
- The supporting infrastructure has been too costly to deploy and maintain;
- There is no efficient network topology to support disconnected users;
- They have failed to address the social concerns of their users.

In the next chapter, we will present the design of a system architecture that addresses these issues. The hope is that the system architecture can help system designers easily build prototype systems to validate the claims in context-sensitive mobile computing.

This chapter has also reviewed a number of active projects in the field of software architectural support for context-sensitive systems. In particular, it was hoped that the review would reveal a number of challenges and common agreements regarding the approach in this area. The components of the existing software architectural support for context-sensitive systems can be summarised in the following figure 3.13:

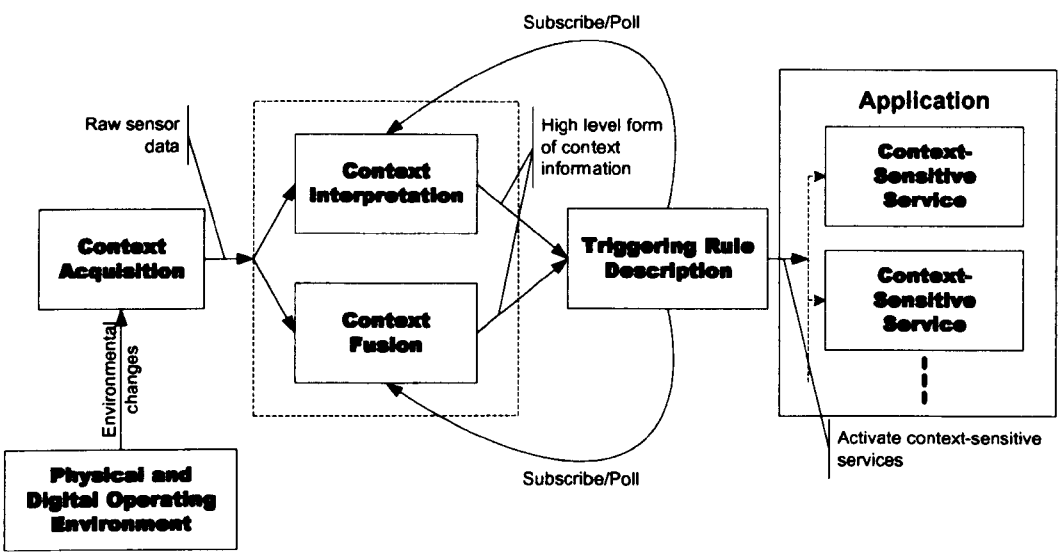


Figure 3.13 Common Agreements on Software Architecture for Context-Sensitive Systems

Context Acquisition

Making computing systems become more sensitive, we need to increase the bandwidth of input channels into computers. One consequence is that the same context can be derived from different sensors. This raises difficulties for developers when the underlying sensors are changed. The aim of this component is to provide developers an abstract view of accessing sensors that are used to obtain context information from the current operating environment. The benefit of this approach is to minimise the impact of sensor change.

Context Interpretation and Context Fusion

The sensed data are needed to be converted into higher level forms of context information before making it available to applications. The idea of this approach is to keep the abstraction of context sensing. For example, applications that need to know their users current location do not need to implement functions to convert longitude and latitude parameters from a GPS receiver or infrared signal into a proper format that suits the application. Again, this resists the impact from the change of underlying sensing technologies. In most situations, it requires several pieces of context information about an entity to determine what it is happening. An application might need to obtain multiple pieces of context information in order to perform relevant context-sensitive services in a situation. This is where context fusion comes into play.

Triggering Rule Descriptions

Context-sensitive applications provide services based upon certain conditions that are predefined by the developers. This component enables developers to determine situations using the context gathered from context acquisition and processed by context interpretation and context fusion components. This also clarifies the circumstances that context-sensitive services should be activated. The reason why this component is outside the application and placed in the software architecture support is that this approach may help non-programmer to author triggering-rules as mentioned in the Stick-e Notes.

Messaging Service Support

Context-sensitive applications need to obtain context information from sensors in order to trigger relevant services to act upon the current situation. The data publishing and subscribing architecture attracts researchers in this field to adopt it in their architectural support. The benefit of this approach is to separate the dependence between applications and sensor components. Following this concept the sensor components simply produce events about the sensed data and can be reused. An application purely plays the role of consumer to exploit the events it is interested and act upon them.

The ultimate goal of a software architecture support for context-sensitive systems is to provide developers a more abstract view of the whole system. It shows them where to start and what are essential to construct this type of system. In the next chapter, we will present the design of a prototyping environment for context-sensitive mobile computing systems. The software architecture of the environment that takes the common agreements into account will be described in Chapter 5. The hope is to demonstrate whether developers can benefit from the previous proposals.

CHAPTER 4
DESIGN OF GLASGOW CONTEXT SERVER – A PROTOTYPING ENVIRONMENT FOR SUPPORTING CONTEXT-SENSITIVE MOBILE COMPUTING APPLICATIONS

In the previous chapters, we indicated that previous research in the field of context-sensitive mobile computing enables systems to take responsibility for partial tasks and actions intended to be performed by the user. In addition, we described the underlying infrastructure of each project in its use of location and environment sensing technologies and end user mobile computing devices. Given these technological innovations it is perhaps surprising that relatively few context-sensitive mobile computing systems have been successfully developed. This possible lack of success can be blamed upon four principle limitations. Firstly, applications have not been integrated with existing mobile computing technology and have relied upon additional hardware such as specialist badges, electronic tags, and other devices to communicate location and environment information. Secondly, the supporting infrastructure that is required to obtain and provide context information has been too costly to deploy and maintain. Thirdly, there are few efficient network topologies to support disconnected users. Finally, they have failed to address the usability and social concerns of their users.

The usability and social problems that might arise from interaction with context-sensitive mobile computing applications are relatively little known so far. Some research has indicated privacy concerns regarding the disclosure of location data when systems exploit it as context information (Want et al., 1992). Others have pointed out that users have to "remember" to perform sensor tasks to keep systems working (Salber et al., 1999). There are reasons that prevent researchers in this field from discovering more information about usability and social issues within this type of system. In particular, the development of context-sensitive mobile computing systems requires considerable engineering skills. The designers who have the necessary expertise to implement these applications often concentrate on innovative system architectures rather than detailed usability and social studies (Burrell and Gay, 2002, Lueg, 2001). Our intention in this research is to build a flexible test-bed for the prototyping and evaluation of human-computer interfaces in context-sensitive mobile computing. It would help interface designers validate some of the claimed benefits of context-sensitive mobile computing systems.

4.1 GLASGOW CONTEXT SERVER (GCS) ENVIRONMENT

We have observed that the use of context information from the previous research projects has mainly focused on location identification and the detection of surrounding resources in the user's vicinity. We, therefore, built a prototyping test-bed to detect a user's location and surrounding objects.

We noted that there are a number of technologies involved in context-sensitive mobile computing systems. These include mobile computing devices, sensing technology, wireless communication technology, and distributed software support, such as server technologies and communication protocols from previous work in this field. Figure 4.1 shows the relationship between each component from previous work in context-sensitive mobile computing.

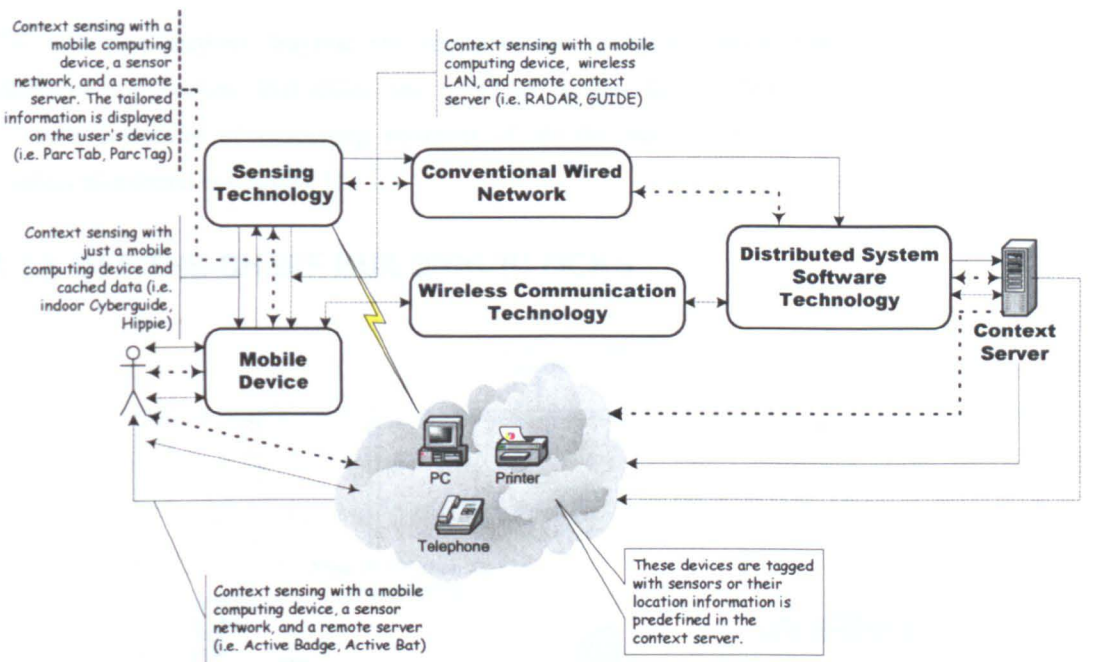


Figure 4.1 Architectures for Context-Sensitive Mobile Computing Systems.

Mobile computing devices can be considered as a link between their users and the physical space which they inhabit. Sensor technology offers mobile devices the capability of measuring their users and environments. Mobile computing environments are rich in context. The environmental and user state are changing over time. Wireless communication provides the connection between mobile devices and the remote server. It enables the systems not only to use cached information on the devices but also dynamically update information as the user moves around her environment. It is important to notice that the hardware and software of mobile computing devices and server machines are varied. The designers have to use heterogeneous software and hardware platforms to support data exchanging between mobile computing devices and remote context servers. The IEEE 802.11 series wireless communication standards are mature and available in the commercial market. A range of middleware has been developed to support context-sensitive interaction (Cheverst, 1999, Michahelles,

2000). Some systems exploit commercial systems such as CORBA (Harter et al., 1999). Others rely upon software that often has benefits in terms of functionality but which is not widely used by others implementing similar systems (Dey, 2000). It also creates problems for anyone who is more interested in the development and evaluation of interaction in context-sensitive mobile computing than they are in the underlying technologies.

We chose web protocols to provide standard data communication in our design system. There are a number of reasons for adopting this approach. Web technology is ubiquitous (Hohl et al., 1999, Kindberg et al., 2000, Leonhardi and Bauer, 2000, Spohrer, 1999). It is possible to obtain access to the web in most geographical locations throughout the world. Web protocols provide a well-understood infrastructure for the designers. This is essential if research projects are to be propagated beyond the laboratory bench into end-user environments.

The following sections describe the design of the Glasgow Context Server (GCS) and how it addresses the concerns that arose from previous approaches. In section 4.2, we demonstrate our system's capability of supporting elements of all the main context-sensitive mobile computing systems mentioned in Chapter 1.

4.1.1 OFF-THE-SHELF BUILDING BLOCKS

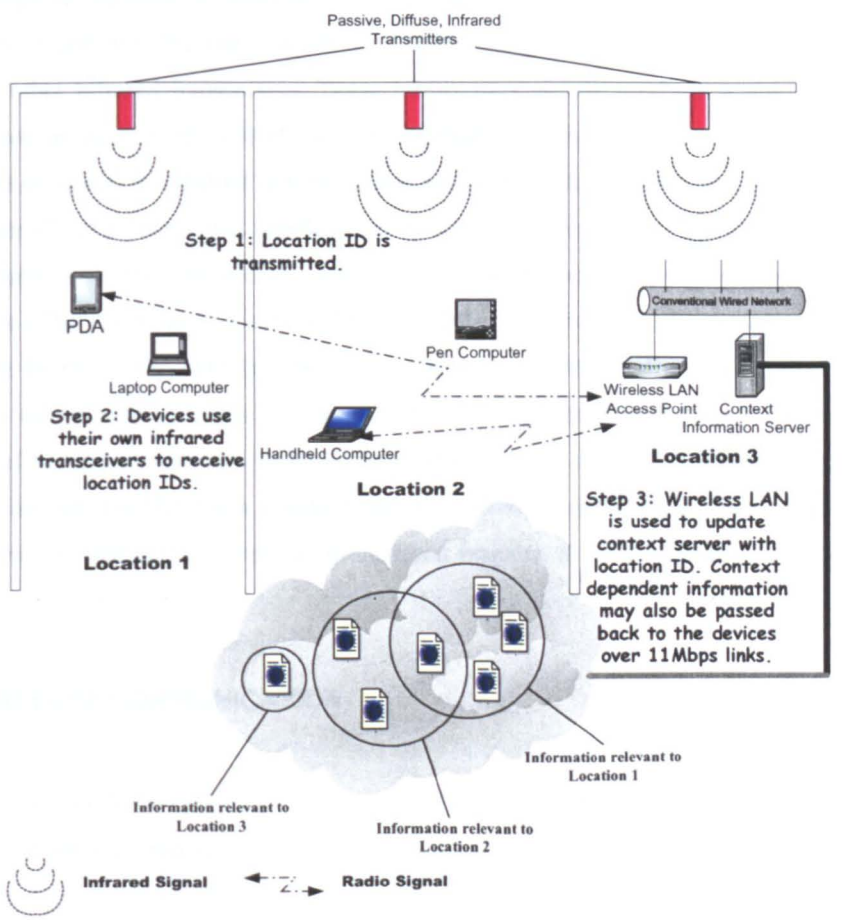


Figure 4.2 The Glasgow Context Server Environment (Johnson and Cheng, 2002).

The GCS was initially designed to provide a low-cost test-bed for the use of indoor location detection in human-computer interfaces in a mobile computing environment. The idea was to provide a means of discovering whether or not some of the claimed benefits for this form of interaction could actually be realised by the evaluation of prototype systems. The focus on low-cost techniques was born from pragmatic constraints. Firstly, we had limited software development resources. Secondly, we wanted to entirely avoid any reliance on esoteric hardware. Thirdly, we wanted to avoid any change to the infrastructure or fabric of the buildings that would house a system. In addition, applications built upon the GCS should be simple to maintain and should not require periodic re-calibration in order to continue to provide accurate location information. The GCS should be modular and easily extensible. That means that the GCS environment should be "future proof" in terms of both hardware and software. Figure 4.2 shows the GCS environment and how it built on previous work. Our initial work concentrated on passive location sensing systems. Later sections will describe how this initial approach has been extended to support the more diverse range of context-sensitive mobile computing systems.

4.1.1.1 SENSING TECHNOLOGY

Rather than adopt the active sensing techniques exploited by the Active Badge project, we follow the indoor Cyberguide approach of utilizing passive infrared transmitters that are mounted in the ceilings of any room or corridor. We use low powered, diffuse transmitters that do not directly detect the presence of other infrared transceivers. These transmitters simply send out a unique identifier. As mentioned, we are eager to exploit off-the-shelf options. In order to reduce costs, these transmitters can be based on domestic infrared remote controllers. The infrared signals can be identified using a range of commercial software that enables mobile devices to detect and replicate the infrared signals emitted by these low cost commercial devices, such as OmniRemote[8], PalmRemote[9], etc. The reason that we chose simply to transmit location identifiers rather than perform active sensing is that we want the system to be entirely mobile. In other words, there is no additional wiring required to install the GCS system into a building. This is motivated by a strong desire to reduce both the cost and the complexity of our implementation. It should be emphasised that the GCS has a modular design that is consistent with both the Bluetooth standard and with differential radio signal detection. These more elegant approaches can simply replace the infrared beacons that we currently use as a pragmatic alternative.

4.1.1.2 WIRELESS COMMUNICATION

Using the GCS, mobile computing devices such as PDAs and laptops can detect their location using existing built-in infrared transceivers. This avoids the external specialist sensors that were a feature of

---

[8] http://www.pacificneotek.com/omnisw.htm OmniRemote™ v1.171

[9] http://hp.vector.co.jp/authors/VA005810/remocon/premocce.htm PalmRemote (R)

the indoor Cyberguide system, Hippie, etc. In its simplest form, the GCS can function without any mobile telecommunication supports. Location dependent information is cached on the mobile computing device. The scenario of using GCS is that as the user moves around the building and infrared signals are detected, applications based on the GCS environment can make use of this data to tailor their presentation of information. Perhaps the most innovative feature of the GCS is that it integrates infrared location detection with a commercial, radio-based, local area network. It is based around 802.11 technology in the current implementation. There are a number of reasons for choosing this system. Firstly, it is simple to install and operate. The user needs simply inserts a PCMCIA card into her mobile computing device and installs the relevant driver. From then on, the system provides the users with a transparent network connection. The user notices no difference between wired and wireless interaction. The integration of radio-based LAN and passive infrared sensing offers benefits that the system can be installed rapidly into environments where it is difficult or impossible to install more conventional systems.

4.1.1.3 WEB TECHNOLOGY

Web technology is used for data communication and software platform building blocks in the GCS. The web offers considerable support for the development of mobile interactive systems based upon GCS environment. Web technology runs on heterogeneous collections of hardware and software platforms (Hohl et al., 1999, Kindberg and Barton, 2001, Spohrer, 1999). The access mechanisms range from PDAs, and mobile phones, to web enabled appliances such as TVs. Servers and end-user browsers have been developed for many different hardware platforms and operating systems. Recent efforts in porting web technology into a small yet growing range of consumer mobile computing devices promise the benefits of exploiting web technology to build context-sensitive mobile computing systems. This development helps avoid many of the overheads associated with supporting an increasingly diverse range of devices. There are further benefits from exploiting the web as a component of context-sensitive mobile computing systems. The relative simplicity of the HTTP protocol and associated addressing schemes has fuelled the rapid expansion of web access. Some researchers also argue that the web provides "just enough middleware" (Kindberg and Barton, 2001). The infrastructure does not assume that devices will all run Java or CORBA. Similarly, the web does not assume the presence of global services. It is possible to access local servers if wider connectivity is denied through accident or design. In the next section, we will describe how the web-based infrastructure helps support disconnected users.

4.1.2 NETWORK TOPOLOGY FOR OFF-LINE SUPPORT

A radio-based LAN provides a mobile computing environment for GCS users. Even with maximal coverage, however, there would be areas where the signal cannot be guaranteed to reach. The physical construction of most buildings creates problems of interference and shielding that cannot be resolved easily. This issue applies both to indoor and outdoor systems, to cellular and satellite systems. The

communication technologies in GCS are built around a radio-based LAN and commercial infrared remote controllers. When the users walk around inside the radio and infrared network coverage, they can access information from the server without any problem. What happens if the users walk into areas where the radio frequency signals cannot reach? Fortunately, a number of features in the architecture of the GCS system enabled us to profit from adversity. The problems of achieving uniform radio coverage forced us to consider the usability issues that arise when people move outside the system. Radio transmitters are expensive but infrared transmitters are cheap. As a result, the GCS system continues to provide context-sensitive services even when users move beyond radio coverage. It is possible to deploy dozens of low cost infrared transmitters that index into pre-cached information on each PDA. The cache is then periodically refreshed when the user moves into radio coverage. Figure 4.3 illustrates an example of the distribution of radio transmitters and infrared beacons.

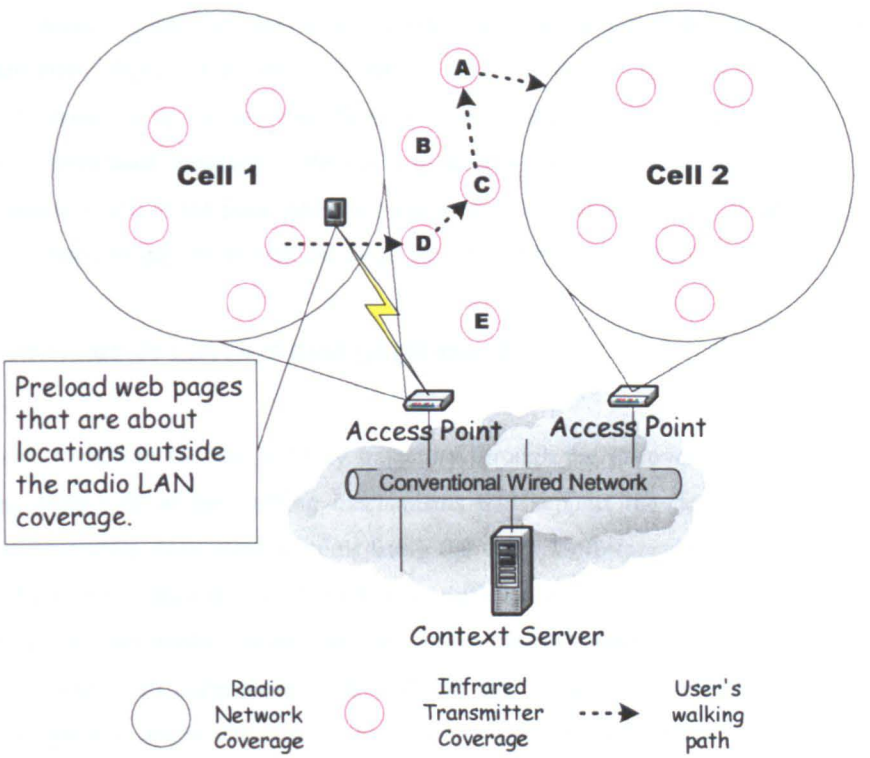


Figure 4.3 A Network Topology for Off-line Support.

4.1.2.1 CACHE MECHANISM

The figure illustrates how a user can walk from radio coverage in cell 1 through an uncovered area to reach cell 2. On their way, they pass infrared beacons D, C, and A. In order for the user to access information about D, C and A, it must be preloaded before the user leaves cell 1. Here is a scenario with more details: A group of web pages can be associated with each of the locations that are represented by the infrared transmitters. Instead of expecting the user to explicitly request the pages associated with each of these locations, the GCS location-sensing system can automatically present

them to the user once the infrared signal has been detected. If the user is within radio coverage, inside cell 1 or 2, then the GCS can automatically request any relevant pages that are not already cached on the mobile computing device from a web server. Otherwise, if the user follows a trajectory that will take them outside of the radio coverage then it is also possible to cache any pages that the designer predicts might be accessed before the user returns to a radio cell. Our approach is to associate web pages about the locations that are outside the radio coverage with the infrared cell near the border of radio transmission. The GCS can automatically pre-cache these web pages to the user's mobile computing device once the user walk into the infrared cell near the border. The web infrastructure offers considerable support for the implementation of this approach. Servers will, typically, provide clients with HTTP headers that specify a cache timeout for each page that a user might request. The header can also indicate when a document was last changed. Context-sensitive mobile computing systems can exploit these attributes of the HTTP in a number of ways. For example, if a user has cached a large document and they are about to leave a radio cell then the browser need only check the header to determine whether or not they will need to download a more recent version of the document. If the current version is up to date then they need not waste critical communications resources in initiating a new download. Similarly, if the cache timeout is greater than the disconnection period then the system need not refresh the page when the user returns to radio coverage. In contrast, servers can use the HTTP header to specify that certain pages should not be cached at all.

4.1.2.2 ENVIRONMENT LAYOUT AND USER MODEL

The GCS must anticipate the user's likely trajectory through the uncovered region. This trajectory plays an important role in the caching mechanisms that support the development of interactive, context-sensitive mobile computing systems using the GCS. Unfortunately, the task of compiling a cache for off-line interaction is complicated. It is difficult to identify all of the potential pages of information that the user might require. Similarly, if location information is used to index into a cache of previous collected web pages there is typically no easy means of determining how long to hold those pages or when to update them as the user moves from one location to another. The traditional model of web-based interaction is almost entirely driven by the user's explicit actions rather than by inferences about their likely needs in a particular location. In some cases, it is possible to use knowledge of the layout of a building to inform the selection of pages for presentation outside of radio cells. For instance, if a long corridor connects two areas of coverage then designers can make strong predictions about the user's possible route and hence their information needs. Alternatively, users can be given directions that explicitly guide their movements. Designers can then predict the information that the system needs to cache as users follow a prescribed route between radio cells (Davies et al., 1999).

However, if the user moves away from an anticipated path then a location sensing system may not be able to provide access to any associated web pages that have not been cached before leaving radio coverage (Cheverst et al., 2000). The system can, however, continue to record information about the

user's route. The system can then offer the user the option of viewing any missing information that is associated with the infrared cells that they have visited once they regain radio coverage. Designers can also exploit this route information to improve the cache content for future users of the system. The existing web infrastructure supports these techniques. However, a number of detailed changes must be implemented. In connected operation, most browsers report a "`146, connection refused`" warning if they cannot access a server. In disconnected mode, the GCS also logs this information to improve cache performance. Further changes must also be made if standard browsers are to support location sensing. For instance, HTTP requests must be automatically generated if the GCS detects an infrared cell. Requests to fill the cache must also be triggered if the users' trajectory will lead them out of a radio cell. Conversely, user input, including requests for web pages that are not in the cache, must be flushed from the cache when they return to radio coverage and the appropriate information returned. The designer's task of supporting such forms of interaction is complicated by a number of pathological situations in which a user might repeatedly move in and out of radio coverage. This is similar to the case where a user hits the "reload" button on the browser repeatedly. This can place a heavy demand upon network resources as the system attempts to update the cache with appropriate web pages. The overheads associated with such movements can be reduced by only retrieving documents whose cache-life has expired. This can be determined using the HEAD component of the HTTP, mentioned in previous paragraphs. It is again important to emphasise that such detailed considerations can have a profound impact on the usability of any location-sensing system that exploits web technology. If these techniques are ignored then there is a danger that the performance of any implementation will be too slow to support the evaluation of context-sensitive modes of interaction. This example also illustrates the need to develop appropriate prototyping environments that might enable designers to explore the human factors issues of context-sensitive interaction without having to face such relatively low-level issues.

As mentioned, the user's cache is updated when the system detects a trajectory that will take them beyond radio coverage. This is done by ensuring that infrared beacons are positioned a short distance inside the perimeter of radio coverage. This distance helps to determine how long GCS has to finish refreshing the information in the users' cache. It is possible to derive a number of equations that support the installation of the GCS network. It should be noted that the following formulae represent gross simplifications. The time taken to transfer location information from the final beacon and to establish wireless communication is factored into the effective data transfer rate. They do, however, illustrate important properties of the general architecture:

$$\textit{Time required to refresh the cache} = \frac{\textit{Quantity of data to be transferred}}{\textit{Effective data transfer rate}}$$

$$\textit{Distance of last beacon from perimeter} = \textit{Time required to refresh the cache} \times \textit{Users average walking speed}$$

The practical development of context-sensitive mobile computing applications has also increased interest in many issues that have not conventionally been part of human-computer interaction, for example, the speed at which people walk and the architectural modelling of locations where the systems is deployed. As mentioned previously, this determines how long the system has to refresh the cache before a user leaves cellular coverage. It is far easier to cache information if users must follow a single path along a corridor than it is to predict information needs in less confined locations.

4.1.3 USABILITY AND SOCIAL CONCERNS

As mentioned, the intention in designing the GCS environment is to provide a low-cost test-bed for context-sensitive mobile computing systems using off-the-shelf components. The hope is that researchers in this field can benefit from the GCS to explore some of the claimed benefits for this type of interaction by the evaluation of prototype systems. In additional to concerns about disconnected users, there are a number of usability and social considerations in the design of GCS. We focus on whether the system is easy to use, rapid deployment, power saving, and privacy.

The reason we chose off-the-shelf 802.11 technology is that it is simple to install and operate. Users often notice little difference between conventional wired and wireless interaction. In addition, the technology can provide 11Mbps connection and up to 57Mbps by IEEE 802.11a. Installing a server is also relatively straightforward. Wireless network interface cards and drivers are currently available for a range of mobile devices such as laptops and PDAs. This eases the overheads for a context server to support a heterogeneous range of end-user mobile computing devices.

Systems that adopt active sensing techniques mean that the sensors actively detect signals from the user's device (Want et al., 1992). In contrast, systems that adopt the passive sensing technique rely upon the user's device detecting signals that are generated from beacons in their environment (Schilit, 1995). Systems that exploit active sensing mechanisms need not require their users to perform explicit sensing processes such as holding a mobile computing device close to the sensing signal transmitter or attaching an electronic tag to the corresponding reader. In contrast, systems that follow the passive sensing approach often require users to perform explicit sensing processes to complete the information gathering processes (Salber et al., 1999).

The integration of a wireless local area network into GCS provides a number of benefits. Not only can the user's mobile computing device detect their location using the passive signals that are transmitted by the infrared technology, they can also exploit the radio-based network to actively communicate their location information back to the context server. This illustrates a crucial benefit of the GCS. The GCS hardware architecture decision avoids any additional wiring or processor support. The user's mobile computing device will only update its location information when the user actually turns it on. There are benefits derived from this design approach. In that, there is improved power saving and

reduced privacy concern. As we shall see, this design decision also helps to avoid many of the ethical issues associated with the systems that gather their users' location information actively. In the case of GCS, the user's device must be switched on and she must explicitly indicate that she wishes to reveal her location information to the context server.

4.2 BUILDING CONTEXT-SENSITIVE MOBILE COMPUTING SYSTEMS BASED ON THE GLASGOW CONTEXT SERVER

As previously noted, the GCS was initially conceived as a location-filtering system. It exploits knowledge of the user's location to tailor the information that is presented to her/him. The following sections describe how the GCS implements two different types of context-sensitive mobile computing systems.

4.2.1 BUILDING LOCATION-DISCLOSING MOBILE COMPUTING SYSTEMS BASED UPON GCS

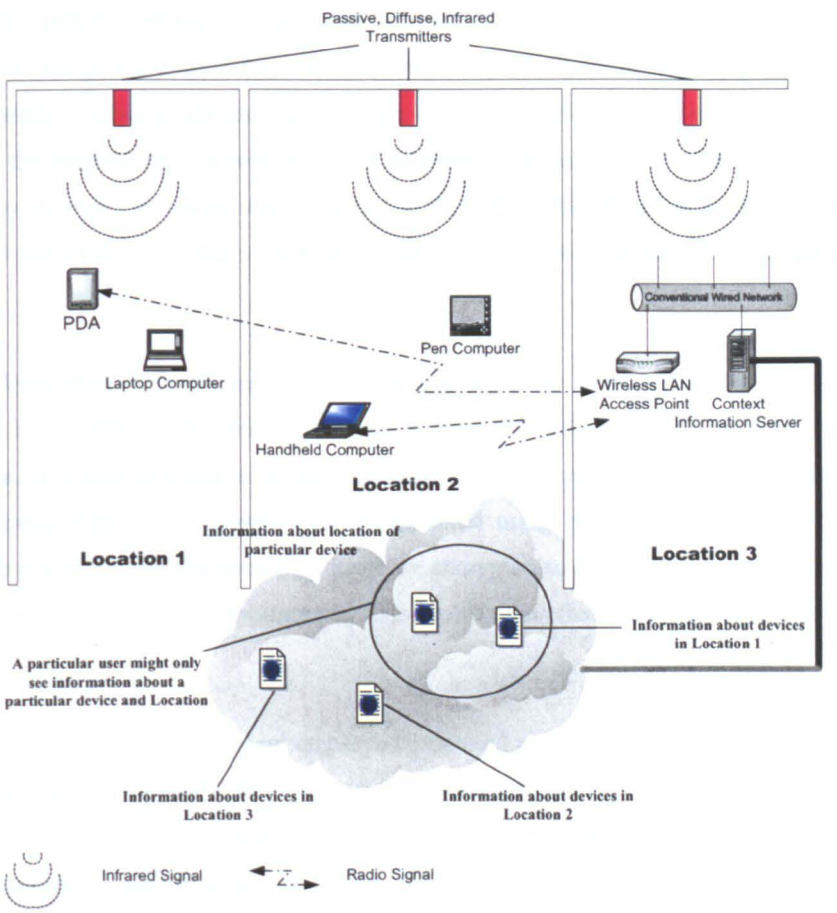


Figure 4.4 Location-Disclosing Mobile Computing System Based upon the GCS.

Figure 4.4 shows how the initial GCS environment must be adapted to support the development of location-disclosing systems. As can be seen, the underlying hardware infrastructure remains unchanged. All of the redesign activities are focused on the web-based infrastructure. In the initial version of the system, changes in the user's location were detected by the infrared transceiver on the user's mobile computing device. These were translated into HTTP requests that were pre-determined by the designers. In contrast, a location disclosing version of the GCS system sends update information to the server every time a new infrared cell is detected. Server side techniques such as common gateway interface (CGI) scripts can then be used to update web pages that disclose the location of the user's device. Such changes are relatively straightforward. It is also feasible to create hybrid systems that combine elements of this approach with the presentation of location dependent information that was described in the previous section. This creates interfaces that are characterized by zone 3 in Figure 1.1 in the Chapter 1. Also although this is still a passive system, we could in the future extend GCS with differential signal processing to obtain active system at no additional cost.

A number of usability issues stem from our use of the web to disclose information about users' location. Firstly, as mentioned previously, because we are using low-cost passive sensors, the user's location will only be updated if she has her PDA or other mobile computing device switched on. One consequence of this is that the location data will not be as accurate as a "true" Active Badge system that continually updates the user's location at all times. We are, however, prepared to accept this limitation given the privacy concerns that have been expressed about these more active forms of sensing. Secondly, usability concerns stem from our use of the web as a platform for location disclosure. In the simplest case, a web page that provides information about the user's location could be retrieved by anyone who has access to the host server. We were, therefore, concerned to introduce access control mechanisms so that a user could explicitly grant her colleagues the right to view this information.

There are at least three different means of restricting access to web pages using standard browser technology. It is possible to force users to provide a password before they can access particular pages of information. It is also possible to setup a filter to restrict access to a server using the client's IP (Internet) address. Finally, documents can be protected using public key cryptography. Both the request for the document and the document itself are encrypted so that the text can only be read by the intended recipient(s). Public key cryptography also enables the client to authenticate the source of the information.

Initial versions of the GCS environment employed password protection to enforce access control. This introduced significant overheads. It proved to be a non-trivial task to set up and administer the privileges that were associated with particular users. This approach is also relatively inflexible. It is difficult to temporarily suspend access privileges if a user wants to prevent disclosure from a subset of their colleagues. Encryption provides a more flexible approach in which temporary changes can be made to the keys that are needed to access particular items of information. This simplifies the implementation of fine-grained access control policies because protection mechanisms are associated

with resources. The converse problem is, of course, that it can be difficult for designers to enumerate all of the potential users who might have the key that is necessary to access an encrypted page of information. Finally, it is possible to restrict access to users associated with a particular internet (IP) address. This offers significant advantages in terms of the usability of any context-sensitive mobile computing system because users are not explicitly prompted to remember passwords and keys. Unfortunately, significant problems affect the two standard means of implementing this approach. Server access restrictions can be placed at the directory level. This is a flexible approach that can be implemented by information providers. However, it can be error prone when individuals forget to remove temporary restrictions or to apply necessary prohibitions. Server access restrictions can also be applied at a global level. This has the benefit of creating clear policies for users to follow. Unfortunately, it lacks the flexibility of more fine-grained approaches. It should also be noted that a number of companies, including Microsoft and Netscape, have launched browser extensions and associated services that provide significant support for the maintenance of access control policies over the web. This area is also a focus for recent work on the future infrastructure of the web (Stein and Stewart, 2001).

The higher-level point is that the lack of fine-grained, web-based access control mechanisms can have a profound impact on users' confidence in location-disclosing systems. Many of the users who participated in the initial design of the GCS system expected to implement a broad range of access control policies. Some individuals wanted to grant their colleagues the right to view their location wherever they went. This led to conflict because other users wanted to deny access to information about people who were gathered in particular locations. For example, the head of one of the organizations that we studied was reluctant to transmit information about workers who might be in his office on confidential or personal matters. Such observations illustrate the way in which the detailed implementation of the underlying web-based infrastructure had to be tailored in response to the observations that emerged from initial usability trials. Access control should not only protect the privacy of particular users, it may also have to be applied to all users in a particular location

4.2.2 BUILDING ENVIRONMENT-SENSING MOBILE COMPUTING SYSTEMS BASED UPON GCS

A number of research groups have identified the potential benefits that might be gained if users are provided with more active, environment sensing systems. These applications go beyond the location-filtering and location-disclosing approaches that have been described in previous paragraphs. In contrast, this new generation of active systems form part of a wider vision in which users are increasingly nomadic but will require continuous access to a range of computational resources "at work, at home, at play". There is also an expectation that they will be able to access these resources in a quick and convenient manner. Users must be able to discover where the nearest resources are to be found. They must then be able to negotiate access to printers, to the Internet and to a broad range of devices in many different physical locations. This work is particularly relevant within the context of

this thesis because some of the proposals in this area are explicitly based on web technology. For instance, the previously mentioned cooltown project associates URLs with objects in the user's environment. Once a mobile device can access these identifiers, they can download and display the pages of information that are associated with these objects. The scenario when a user enters the cooltown environment is described as follows:

"*As you enter a cooltown conference room you can collect the URL for the room from a beacon or tag into your PDA or cell phone. The URL will lead to a Web page for the room giving links to, for example, the room's projector, printer, and electronic whiteboard, as well links to Web-based maps. Workers in the room can also "eSquirt" URLs at the room's "Web appliances" such as Web-enabled projectors or printers. An eSquirt is just a wireless transfer of a URL over a short-range wireless link. Squirting a URL at the projector will project the corresponding Web page, creating a shared Web browser. Squirting a URL at the printer will fetch the Web pages and print them*" (Kindberg et al., 2000).

The quotation shows how environment-sensing systems can be built from the same technologies that the GCS has used to support both location disclosing and location sensing systems. Short-range signals provide the identifiers, or the cooltown URL's, of the objects and people in the immediate vicinity. The information that is associated with these addresses can then be retrieved over longer-range networks.

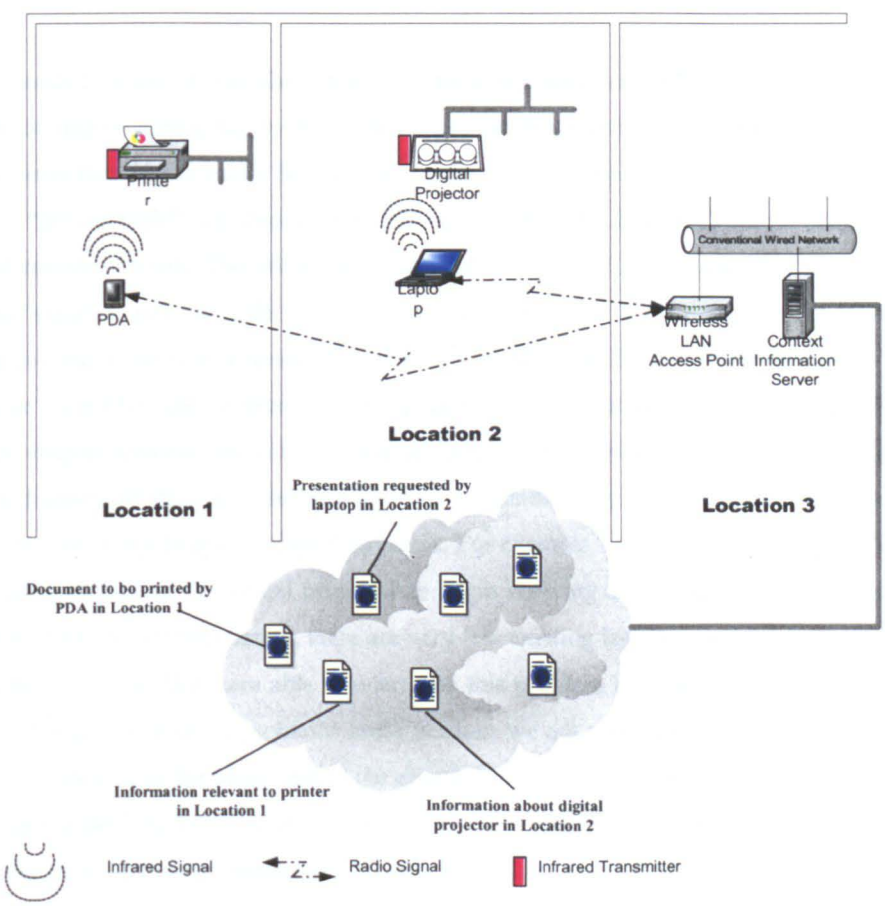


Figure 4.5 Environment-Sensing Mobile Computing System Based on the GCS.

Figure 4.5 illustrates the way in which the GCS environment can implement a variant of the cooltown approach to environment sensing. Instead of associating infrared transmitters with fixed locations, they can be attached to objects including digital projectors, printers and mobile devices. The transceivers on a user's PDA can then detect the signals that are generated by each of these objects. There are two means of accessing the information that is associated with these infrared signals. The proponents of the cooltown project envisage the implementation of a protocol whereby these short-range signals are directly translated into URLs. This is not possible in the current GCS implementation as the infrared transmitters in GCS are simply sending out predefined signals that are entirely different from cooltown's approach. More detailed discussion are presented in Chapter 5.

A number of further limitations frustrate attempts to use the web as a principal component in environment sensing systems. As we have seen, HTTP does not support the fine grained access control that must be implemented to protect users' location information. In environment-sensing mobile computing applications, similar support is required to ensure that mobile users have the relevant permissions to access the resources that they discover on their travels. Similarly, it is often necessary to exclude users from devices that are being used by their colleagues. Otherwise, social collision would happen. For instance, a member of the audience could simply eSquirt (sending infrared signals that contain URLs) a request to change the presentation while the speaker was talking! A further limitation of the web-based architecture is that there is no explicit support for information discovery.

We have intended to use off-the-shelf domestic infrared remote controllers to reduce the costs and complexity of implementing the system. These can be programmed to generate relatively simple numeric patterns that cannot easily be associated with the components of a URL. In consequence, we depend on a "lookup table" that associates a URL with each of the different patterns programmed into the infrared remote controls. The details are described in Chapter 5. When the user's device detects a new infrared signal it uses the table to look up an associated URL. If the pattern is not found then the device can communicate with a remote server over the radio-based LAN to retrieve either the URL, which can be stored for later reference, or the page that is associated with the URL and the infrared pattern. The eSquirt applications that are a strong feature of the cooltown system can be implemented in a similar fashion. In this case, the user's device transmits an infrared signal to the transceiver on another device, such as a printer or digital projector. For example, in Figure 4.5 the PDA in Location 1 might eSquirt a request to the digital projector to begin showing a presentation that can be retrieved from a particular URL. Unfortunately, there are very few existing Internet-enabled devices of the form envisioned by cooltown. We were able to overcome this problem by linking these devices to laptops with radio LAN access. Further problems arose because we are relying on "raw infrared" mode. This is entirely consistent with the ideas behind the eSquirt style of interaction. Users simply fire relatively brief messages at devices to index or reference the remote resources that are to be used. They do not engage in the more complex information exchanges that are associated with the IrDA protocol. We have not, however, developed a protocol that would enable the receiver of the eSquirt to decode the

infrared message and extract an arbitrary URL for a presentation. This remains the subject of on-going research. The implementation of the protocol is relatively trivial. It is far harder to predict the range of commands that should be supported between the user's PDA and devices such as Internet enabled digital projectors. Detail about "raw infrared" and IrDA will be described in Chapter 5.

The fundamental concept behind these sensing systems is that users should be able to query their environment to find out what resources it might offer. Previous paragraphs have described how lookup tables can be used to link infrared identifiers with URLs. The tables can either be cached or can be held on a central server. Neither approach is ideal. If mobile devices cache the lookup table then it can be difficult to arrange access to any devices that users might find in an unexpected location. Conversely, holding these tables on a central server can create problems for users during periods of disconnected interaction. A number of higher level criticisms can also be raised about more ambitious visions of mobile computing. For example, there is a lack of practical evidence to support the claimed benefits for many of the proposed applications. This is a significant issue, given that the mobile Internet has not enjoyed universal success. Strong claims about the utility and usability of mobile technology should be avoided until there is direct evidence of benefit to their intended users. We have, therefore, designed and implemented a number of different GCS applications. The intention is to map out the usability issues that affect a broad range of location-filtering, location-disclosing and environment-sensing systems. This will be addressed in Chapter 6 and Chapter 7.

4.3 SUMMARY

This chapter describes the design of a relatively low-cost context-sensitive mobile computing environment using off-the-shelf components. This is intended to minimize the costs associated with installing the system and is also intended to simplify its maintenance. The GCS successfully integrates an off-the-shelf radio-based LAN with infrared sensors as location or object identifiers that have been a feature of many previous context-sensitive mobile computing systems. This integration offers many benefits. In particular, the decision to avoid fixed network solutions means that the system can be installed rapidly into environments in which it is difficult or impossible to install more conventional systems. The design principles behind this architecture were that it should be low-cost, should be modular and easily extensible and that, most important of all, it should utilize off-the-shelf hardware and software. In Chapter 5, we will describe the technical difficulties that arose during the implementation of GCS.

The GCS provides a prototyping environment that is intended to help HCI designers identify the human factors issues associated with a broad range of context-sensitive mobile computing systems. The GCS exploits domestic remote controls to provide low-powered diffuse infrared transmitters. Depending on the type of interaction that is required these can either be used as beacons to denote particular locations in a building or as tags to help users obtain information about particular objects in their environments. This off-the-shelf approach to hardware design has also been carried over into the

software components. The GCS, therefore, makes extensive use of the web-based HTTP protocol. This offers numerous benefits for the development of context-sensitive interfaces:

- PLATFORM INDEPENDENCE

The web infrastructure is ubiquitous. Both servers and browsers will run on a vast array of devices. This is important for an 'off the shelf' approach to context-sensitive interaction because it should be possible to integrate users' existing PDA's into the GCS environment. The GCS system should also avoid making strong assumptions about the hardware characteristics of the server. The web is also ubiquitous in terms of the communications infrastructures that it can exploit. These range from conventional Ethernet networks through to radio LANs and cellular architectures using the Wireless Access Protocol and similar off-shoots from the HTTP.

- FUTURE PROOF

As mentioned, the off-the-shelf design approach offers the benefit of modularity. The current implementation can be replaced by relevant technologies developed in the future. Most important of all, the web infrastructure is both simple and future proof. This is consistent with the hardware design principle of GCS. It is important not to underestimate the importance of these issues when interface designers attempt to expose the products of recent research to more sustained forms of usability testing.

In the next chapter, we will reviews the technical issues that arose during the development of the GCS. Also, the implementation details of the GCS will be described and discussed. The hope is that our techniques might perform as a blue-print for other designers in this field.

CHAPTER 5
IMPLEMENTATION OF THE GLASGOW CONTEXT SERVER

In the previous chapters we have identified the building blocks for a context-sensitive mobile computing system. We have also introduced the Glasgow Context Server (GCS) architecture that provides a low-cost test-bed for use as an indoor location and environment detection prototyping environment in human-computing interfaces. We emphasised that the intention of building such a prototyping environment is to offer a means of exploring whether or not some of the claimed benefits for this type of interaction could actually be realised by the evaluation of prototype system.

As mentioned previously, the GCS is a prototyping environment for context-sensitive mobile computing. It is built using off-the-shelf hardware and software components. The emphasis on low-cost solutions was born from pragmatic constraints. We had limited software development resources and wanted to avoid esoteric hardware components. This chapter describes the software architecture implementation, which fulfils our design principle and is suitable for dealing with the practical challenges of engineering such a test-bed interface, in particular, the communication between the mobile computing device and the infrared transmitter. The implementation of Glasgow Context-Sensitive Framework (GCSF) that provides the application programming interface (API) for the designers to implement applications based upon the GCS is also in this chapter.

5.1 PRAGMATIC CHOICE OF IMPLEMENTATION PROGRAMMING LANGUAGE

Before a decision can be made as to which is the most suitable choice of implementation, there are a number of considerations that need to be addressed. As we mentioned earlier, there is a vast variety of mobile hardware, ranging from palm or hand-held computing devices to pieces of home appliances. Each device has different characteristics and interfaces. Platform-independent software development technology, therefore, becomes particularly important. A context-sensitive mobile computing application should be capable of dealing with network I/O, serial port communication, and providing user interface widgets. The network I/O API provides the ability to communicate with remote servers that offer services for mobile devices over wired or wireless links. The serial port API provides the ability to do low-level I/O through a computing device's serial ports. This is an important feature that allows a mobile computing device to talk to a variety of sensors. The user interface widgets provide ability to bridge the users and the systems. However, this is a challenging issue in the realm of mobile computing because of its various characteristics. The following paragraphs compare common development toolkits for mobile devices in terms of those requirements.

5.1.1 JAVA 2 PLATFORM, MICRO EDITION (J2ME)

The greatest strength of Java is its portability implemented by its classes being interpreted by a number of device specific virtual machines (Venners, 1999). The Java 2 micro edition (J2ME) is built for small devices that have limited power, network connectivity, and graphical user interface capabilities. It was originally targeted for 16 or 32-bit processors, 16 MHz clock speeds, and devices with 512K or less memory. The J2ME environment includes a Java virtual machine, configuration libraries, and profiles. The virtual machine for J2ME was originally named kilobyte virtual machine (KVM). The configuration is the minimum Java platform for devices that share similar memory size, power requirement, connectivity bandwidth, and user interface options. There are two configurations, Connected Limited Device Configuration (CLDC) and Connected Device Configuration (CDC), for J2ME so far. The CLDC is for devices with up to 512K bytes of memory, for example, mobile phones, point-of-sale terminals, and etc. The CDC focuses on devices with more than 512K bytes of memory. Devices in this category include the set-top box, entertainment system, etc. The profile for J2ME supplements a configuration to provide capabilities for specific devices. The differences between a configuration and a profile is that a configuration must address the lowest similarities for the devices in the same group but a profile concentrates on device-specific factors such as graphical user interfaces, input mechanisms, and database supports. In brief, a configuration and at least one profile are necessary when using J2ME to develop applications for small mobile devices (2002). Unfortunately, J2ME has no cross-platform serial port access. As noted, this is an essential feature for designers to access many sensors if there is no relevant API (Dey, 2000).

5.1.2 WABA

Waba defines a language, a virtual machine, a class file format, and a set of classes (Wabasoft, 2001). Programmers can write Waba programs using Java development tools. In most cases, programmes written in Waba can run on Java Virtual Machines without trouble but the reverse may not be true. Programs written in Waba cannot exploit the total set of Java classes. This is because the syntax, class file, and bytecode format of Waba is a subset of Java's. Also, Waba comes with a set of bridge classes that allow Waba programmers to run anywhere Java is available. For instance, Waba programmes can run as Java applets and standalone applications. Waba applications can only use the classes in the Waba software development kit (Waba SDK). The Waba SDK contains classes for graphics, images, serial port access, database access, sockets and other system functions.

We chose Waba to implement the Glasgow Context Server (GCS) for a number of reasons. Firstly, it is free yet substantial. Secondly, it offers cross-platform support so that applications written in Waba can run on Palm OS, Windows CE or Pocket PC devices as well as java-enabled devices. Ideally, a program written in Waba would execute exactly the same way on all the above mentioned platforms. Thirdly, it provides a relevant application programming interface (API) to access the serial port on

mobile computing devices. This is an important feature for us to interface between mobile devices and a range of sensors.

In the next section, we describe the practical challenges of engineering the GCS, in particular, the interface between off-the-shelf mobile computing devices and sensors.

5.2 REPRESENTING PHYSICAL PLACES AND OBJECTS USING INFRARED SIGNALS

The GCS provides a test-bed that creates a virtual connection between the user, physical places, objects, and digital information services. It focuses on the environmental interaction such as obtaining context information from the user's surroundings including the user's location and resources. A key challenge is how the mobile computing device acquires and interprets context information from sensors. The technological burden such as sensor installation, sensor data format, sensor status change notification can be released from the previous experience of using the sensor. For instance, the designers can take advantage of the existing API for a particular sensor to build applications based upon it. In contrast, the GCS designers have to consider how to communicate with the sensor and obtain the relevant data if the API is not available. Previous researchers have identified that the burden can occur either at sensor or application level (Dey, 2000). They argued that it eases the designers' burden to implement the process of converting sensor data to the relevant forms for responsive applications if the sensor can convert the sensed data by itself. As mentioned previously, the design principle of GCS is to exploit low-cost off-the-shelf components. We, therefore, use domestic infrared controls from television and audio systems to generate location and object identifiers as context information. The identifiers can be read when the mobile devices carried by the users enter the signal coverage of the infrared remote controls. The domestic infrared remote controls used in the GCS can be mounted on the ceiling or attached to physical objects. The location context information is the location of the installed infrared remote control, and object context information is the object that is attached to the infrared remote control. Unfortunately, there is no widely available APIs for domestic remote controllers. We, therefore, had to develop code to read the infrared signal from the remote control and interpret it into a readable data format using hardware on PDAs. Hence, a group of web pages can be associated with each of the locations and objects that are represented by the infrared signals. The following paragraphs review the technical challenges that arose during the development of the GCS and the approaches we took to address any conflicts.

5.2.1 THE TECHNICAL CHALLENGES FOR INFRARED SENSING

As noted previously, the GCS exploits off-the-shelf low-cost domestic infrared remote controls as location and object beacons. Ideally, the mobile computing device's built-in infrared port can be interfaced directly to the receiver that understands the infrared signals from most of domestic infrared

remote controls. This minimises requirements for additional intermediate hardware. Unfortunately, there are two technical obstacles to communication between these infrared remote controls and mobile devices' existing transceivers. Firstly, the protocols and standards used by the mobile computing device's built-in infrared transceivers are entirely different from the physical and data link layers used by infrared domestic remote controls. Secondly, the communication behaviour such as transmission range, coverage, and connection path are also different between domestic infrared remote controls and mobile computing devices' built-in infrared transceivers. The following sections describe the characteristics of these two different forms of infrared communication. We want to have the infrared transceiver on the mobile computing device recognize the signals from a domestic infrared remote controller.

5.2.1.1 CONSUMER DOMESTIC INFRARED REMOTE CONTROLS PROTOCOLS VS. IrDA PROTOCOLS

Consumer infrared domestic remote controls utilize pulse, space, shift encoding for infrared transmissions. They are different from the Infrared Data Association (IrDA) protocol that is exploited by infrared transceivers on most mobile devices. This explains why the indoor Cyberguide system was forced to exploit additional hardware to enable its users to detect infrared signals from domestic infrared remote controls (Long et al., 1996). The details can be found in Chapter 3. The following section describes the differences in terms of the physical layer, which includes encoding/decoding method and data frame format, between domestic infrared remote controls and mobile computing devices with a built-in IrDA compliant infrared port.

Domestic Infrared Remote Controls

How does a domestic infrared remote control work? Each button on an infrared remote control is associated with a specific infrared code pattern. When a user press a button on a remote control, the chipset in the remote control senses the connection and identifies which button is pressed. The transistors amplify the signal and send it to the LED. The LED translates the signal into infrared light. The remote control transmits a specific code that informs the receiving unit to carry out a particular command.

The infrared signals emitted from remote control are modulated in order to filter out interference from ambient light and other infrared sources. A 38 ~ 40 kHz modulation is used in most consumer infrared remote controls (Technology, 2001). The modulated infrared signal consists of a digital signal (device address and command code) and a carrier. Therefore, the corresponding receiver can be adjusted to the specific carrier and reject any other infrared signals that do not have the proper carrier frequency within the sensor coverage. This is fortunate because it reduces problems of signal overlap in the GCS. The digital signal is overlapped onto the modulated infrared signal. A logical one is represented by a burst of oscillations. The resulting signal is the carrier signal being activated and deactivated in

accordance with the specific command code sequence. Figure 5.1 illustrates the concept of a resulting infrared signal from a domestic infrared remote control.

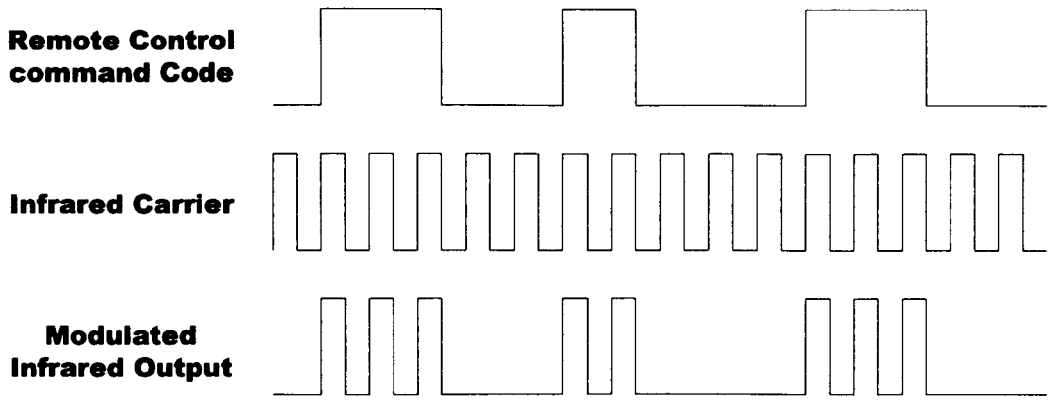


Figure 5.1 Modulated Infrared Signal with Command Code

Manufacturers of domestic appliances have exploited three different means of encoding infrared signals to implement the data link layer in consumer devices: (1) Pulse-Width-Coded Signals, (2) Space-Coded Signals, and (3) Shift-Coded Signals (INC.).

1. Pulse-Width-Coded Signals

This encoding method alters the length of pulses to code the information. If the pulse width is short (roughly 550us) it can be regarded as a logical zero or a low. If the pulse width is long (roughly 2200us) it can be regarded as a logical one or a high. Figure 5.2 illustrates that each bit to be sent is encoded by a low level of duration T followed by a high level of T represents a logical zero and 2T represents a logical one.

2. Space-Coded Signals

This encoding method is also known as REC80. In contrast to previous encoding method, this method is to alter the length of the spaces between pulses to code the information. If the space width is short (roughly 550us) it can be regarded as a logical zero or a low. If the space width is long (roughly 1650us) it can be regarded as a logical one or a high. Figure 5.2 illustrates that each bit to be transmitted is encoded by a fixed high level duration succeeded by a low level of T represents a logical zero and 2T represents logical one. A logical one takes longer to be transmitted than a logical zero in this encoding method.

3. Shift-Coded Signals

This encoding method is also known as RC5 or biphase code. This coding method has a consistent duration in each bit. The logical value is determined by the transition in the middle of the duration interval. A logical zero is encoded by a high to low level transition and a logical one is encoded by a low to high level transition. An additional transition is needed to take place at the

beginning of each bit in order to set the proper start level. This additional transition is considered as a "Header" and it usually consists of a long pulse transmitted before the subsequent serial bits. An example of RC5 encoding code is also shown in figure 5.2:

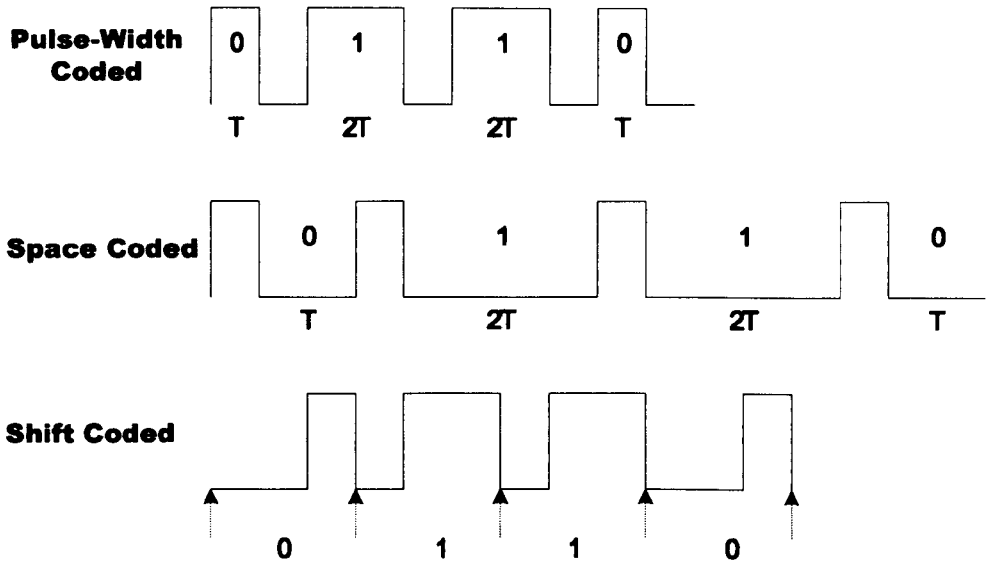


Figure 5.2 Pulse-Width-Coded, Space-Coded, and Shift-Coded-Signals

As previously mentioned, a domestic infrared remote control sends a string of signals when a button is pressed. The first part of the information in the infrared data string normally contains a burst of high-level ones in order to alert the corresponding receiving device to be ready to respond to the following command. The succeeding signal burst is the address of the specific receiving device and the command that the intended receiving device should respond to. The command repeats itself till the button on the remote control is released and a string of code is transmitted to inform the receiving device to stop executing the command. In brief, the string of signals contains information about the key pressed, the address of the receiving device that should respond to the command, and the command. This is implemented by the protocol layer that divides the transmitted infrared code into three parts, header, address, and command.

Remote controls that exploit pulse-coded encoding, format each data packet with a 14 bits data string. It includes a header ($H_1 \sim H_0$), which wakes the receiving device to act upon the following command, an address ($A_4 \sim A_0$), which indicates the receiving device, and a command ($C_6 \sim C_0$). The REC80 (space-coded) control has a format of 32 bits per data package. The first 16 bits represents receiving device code and the reminding 16 bits represent the command code. The second half of each 16-bit set is the complement of the first part. The RC5 (shift-coded) control is formatted as 14 bits per packet. The first two bits, S_1 and S_0, allow the receiving device to tune and synchronize itself. The following bit T toggles with every keystroke. The subsequent five bits ($A_4 \sim A_0$) describe the address of the receiving device, which should respond to the commands sent from the remote control. The rest of the bits ($C_5 \sim C_0$) are the content of command (Mohamed, 1998).

Infrared Data Association (IrDA)

The Infrared Data Association (IrDA) is an industry-based group spearheaded by HP, IBM, and Sharp since late 1993[10]. The aim is to promote an industrial standard for low-cost, short range, cross-platform, and point-to-point two way infrared communications at a wide range of speeds. The standard has been implemented on various platforms, in particular, the infrared transceivers on most mobile computing devices. The following section describes the IrDA physical layer and IrCOMM. The aim is to identify the reasons that prevent the IrDA transceiver equipped mobile computing devices from recognizing the infrared signals transmitted from domestic remote controllers.

PHYSICAL LAYER

The IrDA physical layer defines a standard for the infrared transceiver, encoding/decoding, serializer/deserializer, and a framer (IrDA, 2001). In the case of sending out a string of data, firstly, the framer assembles the data into IrDA frames, secondly, the serializer converts the IrDA frames from bytes to bits, thirdly, the encoder encodes the data bits in the format of a serial digital signal with a specific timing between pulses, and finally, the digital signal is converted to an infrared signal by the transceiver. The reverse takes place in the case of receiving data.

The IrDA physical layer defines the link length between two IrDA devices to be from 0 to 1 meter range within 30 degree cone coverage. IrDA 1.0 defines speeds between 9600 bps and 115.2 kbps; IrDA 1.1 includes 57.6 kbps, 1.152 Mbps and 4 Mbps. Speed below and including 115.2 kbps is called serial infrared (SIR). Speed above 115.2 kbps is called fast infrared (FIR). For the transmission rate of 9.6 k, 19.2 k, 38.4 k, 57.6 k or 115.2 kbps operations, a start '0' bit and a stop '1' bit is added before and after each data packet. This is the same format as used in a traditional universal asynchronous receiver/transmitter (UART). A UART can translate between serial and parallel data. For instance, a UART converts parallel data fed from the CPU on the PC's system bus into serial data for transmitting and does the reverse on the received serial data (Axelson, 1998). The bit stream from the transmitting device's UART is encoded by a pulse-width modulator. The minimum pulse width can be as low as 1.41 us. However, instead of utilizing the non-return-to-zero (NRZ) signal format adopted by the UART, a method called return-to-zero-inverted (RZI) is used, where a zero is encoded as a single pulse of 3/16 of a bit cell, and a one is encoded as the absence of such a pulse. The encoder keeps the pulse width less than 3/16 of the bit-rate period improve power saving in the transmitting LED. Most UARTs have a 16x clock, therefore, it is simple to count three clock cycles to encode the transmitted data and stretch the received data with 16 clock cycles. The edge detector and pulse-width demodulator in the receiver section must decode or stretch the pulses in the incoming bit stream in order to present it to the UART in the receiving equipment.

---

[10] http://www.irda.org

In IrDA version 1.1, the transmission rate is confined to begin negotiation at 9600 bps and then at up to 4Mbps. Unlike the RZI modulation exploited in the transmission up to 115.2kbps, a 4Mbps connection adopts a four-pulse-position modulation (4 PPM) scheme. The 4 PPM encodes the data bit stream for transmitting in the format of a "data bit pair" (DBP) data frame. Each DBP contains two data bits with the duration of 500 ns. There are four states (00, 01, 10. 11) for a combination of two data bits. Therefore, a DBP is divided into four 125 ns time slots to represent each data bit pattern. The demodulation on the receiving device can identify the bit pattern by the location of the signal pulse in a DBP within the 500 ns period. Figure 5.3 illustrates the NRZ, RZI, and 4 PPM modulation.

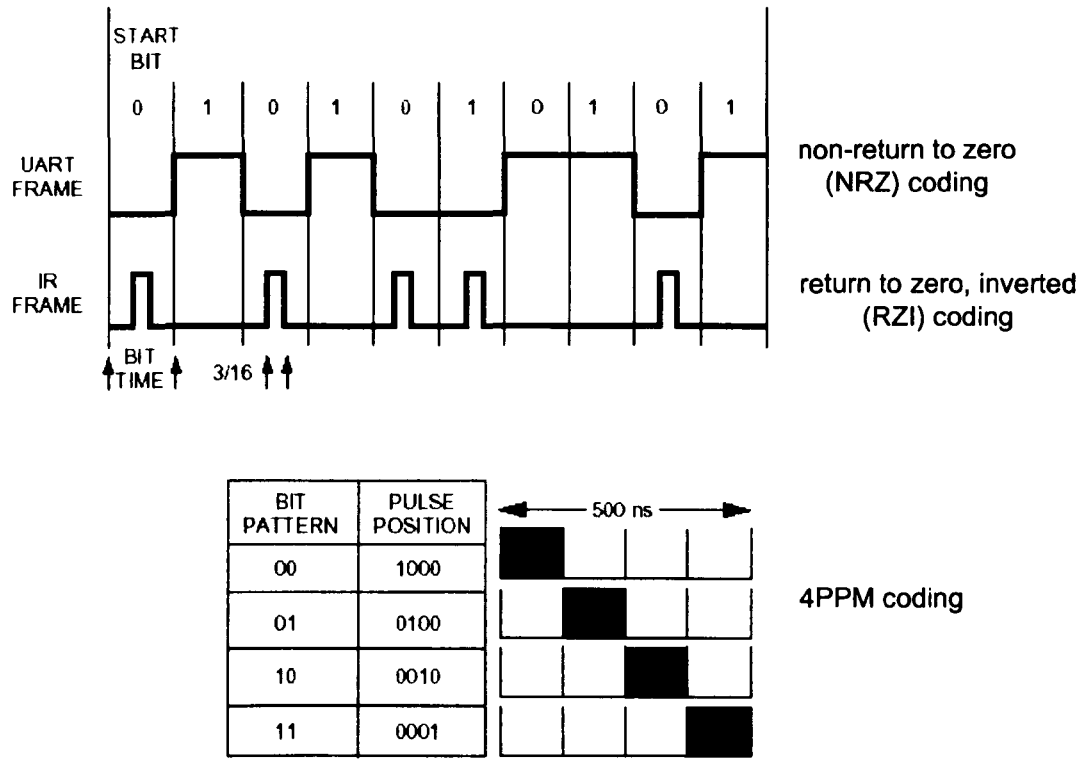


Figure 5.3 NRZ, RZI and 4 PPM Modulations (Source: (Technology, 2001))

The IrDA physical layer defines the synchronous data frame format for transmission at 576 kbps and 1.152 Mbps with two beginning flags, an address field, a control field, an information field, a frame check sequence field and an ending flag. The transmission rates from 9600 kbps to 115.2 kbps are formatted with an asynchronous data framer that consists of an additional beginning of frame byte, a beginning of frame byte, an information field, a frame check sequence field, and an ending flag. For the transmission rate at 4 Mbps, the data frame is formatted as a preamble field, a start flag, a 4 PPM data symbol (DD) field, and a stop flag. The preamble filed emerges from the 4 PPM exploited at the 4 Mbps transmission. It is specially designed for the receiver to setup a phase lock. The information field mentioned in each data frame format is defined as IrLAP payload data. The detail should refer to the IrDA IrLAP specification (IrDA, 2001, Technology, 2001).

IrCOMM

In IrDA, an optional protocol called IrCOMM provides a transparent runtime environment for applications that talk to other devices via serial and parallel ports using infrared communication (Suvak et al., 1995). By making IrDA protocols accessible via these APIs, many existing applications including printing can run over an IrDA infrared link without change. This protocol is a good potential means for IrDA devices to obtain infrared signals from domestic infrared remote controllers.

However, IrDA infrared communications differs significantly from other forms of serial and parallel communications. Serial and parallel communications methods can send streams of information in both directions at once, because there are typically multiple wires. With infrared, there is the only one wire, that is, the infrared path. The IrCOMM standard was developed to solve the problems describe above and allow legacy applications to be used over infrared. There are two key feature of IrCOMM described as follows.

IrDA protocols send data one way at a time. If a device tries to send data and listen for data at the same time, it would "hear" itself and notify the device that it wants to communicate with. The way IrDA devices achieve two-way communication is to take turns. These take place at least every 500 milliseconds, and can be made more frequent as necessary. This latency makes it impossible to perfectly emulate the wired COMM environment. All of the information carried on multiple wires must be carried on the single infrared path. This is accomplished by subdividing the packets into data and control parts. In this way, a logical data channel and control channel are created, and the various wires can be emulated.

IrCOMM makes IrDA look like serial and parallel media that do not have automatic negotiation of best common parameters and a "yellow pages" of available services. For instance, if a word processing application wants to print via infrared using IrDA protocols, an application must first discover and locate the device, the printer, in infrared transmission range, and then check the printer's information service access (IAS) before connecting. Since the word processing application (a legacy application) knows nothing about this, IrCOMM maps these operations into normal COMM operations so that it is completely transparent.

5.2.1.2 COMMUNICATION BEHAVIOUR

A domestic infrared remote controller emits an infrared beam in only one direction in a low-speed burst for a transmission range of up to 10 meters (Technology, 2001). As mentioned previously, IrDA 1.1 physical link layer specification facilitates a two way communication in a transmission rate up to 4 Mbps with effective communication between two devices of up to one meter. Some manufactures claim a transmission range up to 3 meters (Technology, 2001). Normally, a domestic infrared remote control emits an infrared light beam with a wider cone angle than the IrDA devices. The receiver is

designed to detect infrared signals from various directions such as reflections at the ceiling and at the walls with a wide field-of-view photodiode. This leads to improved ease-of-use. The transmitter signal can be received almost everywhere within a room space even if the line-of-sight between the transmitter and the receiver is blocked. Therefore, the infrared front-ends do not have to be carefully aimed to each other. This is difficult for IrDA devices to achieve due to the different methods of use, in particular the requirement for reliable, high-speed data transfer.

5.2.2 APPROACHES TO MAKING MOBILE DEVICES RECOGNIZE INFRARED SIGNALS FROM DOMESTIC REMOTE CONTROLS

To implement the infrared sensing in GCS, what we need is to obtain a consistent data pattern converted from the infrared signal emitted by remote controls using the IrDA transceiver and relevant driver or software protocol. As previously mentioned, the standard for domestic infrared remote controls differ from IrDA in terms of the physical and data link layers. Infrared signals emitted from an infrared remote control cannot be recognised by the IrDA transceiver on most mobile devices due to the rather different nature of modulation, data frame format, and encoding/decoding method. IrDA aims at high communication rates between the compliant devices. In contrast, domestic infrared remote controllers work at low-speed, long distance, and near omni-direction for ease of use. From the previous paragraphs, the difference in operating wavelengths between the two different types of communication standard is the first obstacle to be overcome in enabling the different devices to talk to each other. The operating wavelengths are affected by the modulation technique exploited in the infrared transceiver device. The resulting effect is to reduce interferences from other infrared sources. The wavelength of signal emitted from domestic infrared remote controls ranges from 900 ~ 950nm and IrDA confines the working wavelength from 850 ~ 900nm (IrDA, 2001). When a domestic infrared remote control fires a signal to an IrDA receiver it results in signal attenuation due to the filter in the receiver, which cuts off the signal from the remote controllers. Therefore, the link distance would be less than the normal transmission range of a domestic infrared remote control (IrDA, 2001, Technology, 2001). The second issue with which we need to be concerned is the modulation in an IrDA transceiver and a domestic infrared remote control. The IrDA 1.0 supports data rates ranging from 2.4 ~ 115 kbps, and 4 Mbps is achieved by IrDA 1.1. This includes the operating frequency, 38 ~ 45 kHz, of most domestic infrared remote controls. We can set the baud rate of the IrDA transceiver in a mobile device as low as 2.4 kbps. As mentioned, the UART on most computing devices typically utilizes a receive clock that is sixteen times the bit frequency, so the 2.4 kbps rate should have 38.4 kbps receive clock (2400 (baud rate) × 16 (clock) = 38.4 K) (Technology, 2001). This setup would make the infrared signal from a domestic remote controller go through the filter in an IrDA transceiver. However, the received data varies even though the same button on the remote control is pressed. This is due to the IrDA protocol attempting to interpret the signal with its data frame format, and to extract the data from the frame packet. We have employed two approaches to making the communication between domestic infrared remote controls and IrDA transceivers on most mobile devices meet the requirements of the GCS.

5.2.2.1 SPACE CODED

The REC80 mentioned previously varies the length of the signal spaces between pulses. Inspired by the REC80 coding method for domestic infrared remote controls, we developed software which makes a timestamp once the signal bit stream from a remote control hits the IrDA transceiver on a mobile computing device. We know that when a button on a domestic remote control is pressed the remote control emits a series of infrared light flashes that can be decoded as start bit, a header, a device address, a command, and a stop bit. However, this approach does not decode the incoming signal stream from the remote control and simply monitors the time slots between two series of infrared flash light. On the other hand, we can monitor the frequency of a button press on the remote controller. The time slot is controllable to be unique. Each unique time slot can be exploited to represent physical places or objects where the infrared remote control is positioned. A side effect of this approach is that of conserving the power because the infrared remote controller need not keep emitting the signal, but operates at adjustable power on and off frequencies.

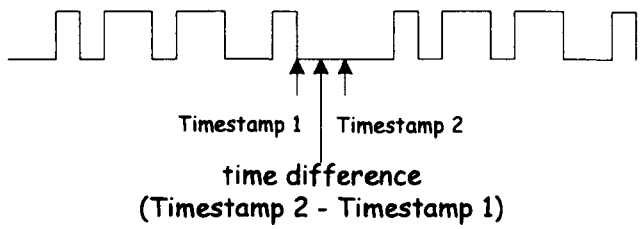


Figure 5.4 Space Coded Approach.

Our initial implementation was based on an Intel Pentium II™ laptop PC with a built-in IrDA port and running Microsoft Windows® 98 operating system. The IrDA communication protocol stack has already been embedded within the operating system. As mentioned, IrCOMM emulates a serial port through an infrared transceiver. This approach can be implemented by using the Java communication API, which is a standard extension and available since Java 1.1. It allows Java standalone applications to access information from the serial and parallel ports of the host computer (Harold, 1999). The Java Communication API provides programmers with a server-based/event-based model to access the serial and parallel port. For instance, they can develop programs to "wait" for incoming information from serial or parallel ports such as a network connection request, paper-empty notification from a printer, and etc. When there is a state change at the monitored serial or parallel port, runtime support issues a serial or parallel event to the registered port listener. In brief, there are three steps to monitor the events from a serial or parallel port using the Java API. Firstly, implement the port event listener interface. Secondly, register the port listener to the monitored port. Thirdly, indicate the occurrence of the event type of the port. The Java Communication API provides ten possibilities for a serial port event types and two possibilities for a parallel port. The detail of each possibility is referred to in (Harold, 1999). We exploited the "`SerialPortEvent.DATA_AVAILABLE`" event type to implement this approach. When there has been data detected at the serial port, the runtime fires a

serial port event and executes relevant procedures. In this case, a timestamp is made. The programme can obtain the time slot between two pulses by subtracting the former from the later timestamp.

Unfortunately, a number of issues arise in this approach. The domestic infrared remote controller has to be altered to generate a specific signalling frequency. However, as mentioned, we are eager to use off-the-shelf components and entirely avoid any reliance on esoteric hardware. This approach violates the design principle of the GCS. In addition, the Java virtual machine ports on our expected Windows® CE and Palm OS mobile computing devices do not all provide serial or parallel port APIs. Waba, however, provides a serial port API with limited support. It does not have the server-based and event driven model used in the Java Communication API. The Waba serial API does not always support "open" and "watch" on the serial port. Instead, it simply provides "open", "close" a serial port, and "read" and "write" data stream. Although the timer control provided by Waba can be exploited to poll the monitored serial port at a specific frequency, the synchronisation problem between the mobile device and the altered domestic infrared remote control may prevent the correct signal from being interpreted. The polling mechanism cannot guarantee the correct signalling frequency from the remote control due to its on/off frequency. It may not be able to receive any data from the infrared transceiver if the polling frequency is the same as the signalling frequency on the remote control. Therefore, programmers cannot develop programs that monitor the event at the serial port using Waba. This makes this approach infeasible.

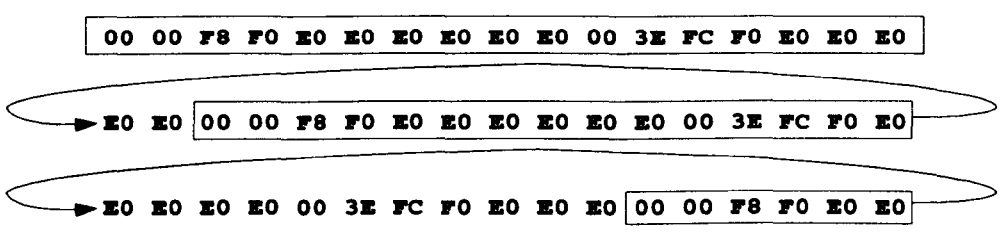
5.2.2.2 RAW INFRARED MODE

According to our findings, the use of raw infrared mode is only feasible on Windows® CE (recently called Pocket PC) platform. The tight binding of the IrDA protocol with Palm OS makes it impossible to exploit raw communication through the infrared port. There are at least four types of communication port implementation in most Windows® CE mobile computing devices: hardwired, spare, IrCOMM, and the infrared communication port. The hardwired communication port requires a null modem serial cable that connects to the host device via the serial port. The spare communication port is designed for a PCMCIA type modem port. IrCOMM, as mentioned, is the IrDA infrared communication port that emulates a communication port and offers point-to-point communication with error checking and data integrity. The infrared communication port, also known as raw infrared port, supports no high level communication protocols. It is possible to use the raw infrared port to receive and analyse the infrared signal from a domestic infrared remote controller, without conflicts emerging from the different encoding/decoding methods mention above. We, therefore, developed software that temporarily replaces these IrDA protocols with the "raw infrared mode". Using raw infrared mode supports the transmission of infrared location and object information and does not force users to carefully position their mobile computing devices within the IrDA's one meter effective transmission range. Similarly, they do not need to discover and negotiate the communication rate with an IrDA device in order to establish the link between each other. Programmers can access the infrared port as a normal serial port with an infrared transceiver attached. In this case, programmers have

greater control of the infrared communication. However, they have to deal with communication issues such as signal collisions and broken links due to something coming in between during a data exchange, when they adopt a non-IrDA compliant method.

In this approach, we simply tapped down a button on the domestic infrared remote control so it keeps generating consecutive signals. The programme we developed to run on the user's mobile computing device simply polls the infrared port to detect and interpret the signals. The signal detection and interpretation task on the mobile device can be easily resumed, instead of initiating the IrDA handshaking and negotiating processes in order to recover the previous failed link if some obstacles come between the two communication devices. A particular concern about the way we use the domestic infrared remote control is the duration of the output signal once a button on the remote control is pressed. Some more advanced models are equipped with power saving control. The infrared signal is only sent for a specific duration no matter how long the button is held. This is an obstacle to our constant polling of the mobile device's built-in infrared port for signals from the domestic infrared remote controls. We intend to exploit the infrared signals to represent physical places and objects. There is no chance of detecting and interpreting the infrared signals in the relevant form of location and object information, if the remote controls cannot emit signal constantly to the mobile devices.

In order to exploit the unique infrared signal emitted from the remote control for our purpose of representing physical entities, we developed a programme, IRTest, which implements the raw infrared mode approach and identifies incoming infrared signals from the mobile computing device's built-in infrared port. Using this program, the raw infrared port needs to be assigned and is then tuned to the baud rate close to the clock frequency used in the remote control. Once the start button is pressed, the program starts to monitor the raw infrared port discretely with a specific polling frequency and converts the incoming signals into hexadecimal value for display. A number of issues arose from the development of this program. Firstly, the raw infrared port number may vary with the model of the mobile computing device. There is no standardised raw infrared port number; instead, the manufacturers have taken responsibility for their assignment. Designers can find the raw infrared port in the registry: `\Comm\IrDA\` key under `HKEY_LOCAL_MACHINE` (Boling, 2000). Secondly, the baud rate needs to be configured in order to detect the signals from infrared remote controls precisely. As described previously, 2400 bps is suitable for picking up most of the signals from domestic infrared remote controls. Our initial experiment was based on the HP Jornada 680 hand-held computer and the KONIG IR 9441 domestic infrared remote control. The following shows a partial sample of the hexadecimal data received from the hand-held computer's built-in infrared transceiver when holding the button "1" on the remote control:



It seems like the same button generates inconsistent output. However, a pattern is present and can be obtained with further analysis. There are a number of reasons for this. The way we use the domestic infrared remote control is to tape down a button on the remote controller and force it to keep emitting signals. However, the built-in infrared port on the mobile computing device is not activated all of the time. The incoming infrared signals from the remote controller may be intercepted at different sections.

Also, as mentioned, the Waba serial API does not provide a server/event-based model of accessing a serial port. The task of monitoring a serial port with an effect similar to a server model approach can be done by exploiting the "timer" method provided by Waba. For instance, we set the timer to 800 milliseconds and timeout to 500 milliseconds in the IRTest so the programme polls the serial port every 0.8 seconds and holds the control of the port for the of 0.5 second duration. Due to the nature of the raw infrared port approach there is no protocol such as data frame check support in the IrDA to guarantee data integrity. The program does not discriminate which point in the output signal cycle is intercepted. It is possible that the signals are intercepted halfway through their transmission. We need to implement an identification search mechanism that treats the above sample data as the same pattern. The details are described section 5.3.1.2.

5.2.3 INSTALLATION OF THE DOMESTIC INFRARED REMOTE CONTROLS INTO THE ENVIRONMENT

The shortcoming of infrared sensing using the mobile computing device's built-in infrared port is that it does not support omni-direction/diffuse sensing but confines it within a ~30 degree cone. This cannot be overcome because of the specially manufactured hardware that fulfils the IrDA physical link layer specification. This gives rise to a number of usability issues. Users are forced to look for infrared transmitters installed within the environment and to direct their mobile computing devices' infrared port to the transmitters in order to detect the location and object information. This may not much impair users wishing to fulfil a task such as viewing information about physical entities. Users always have to approach the places or objects equipped with infrared transmitters and direct the infrared port on the device to trigger the context detection. This, however, may affect the way we install infrared transmitters. The signal transmission range may be limited in some mounting options. These infrared transmitters have to have flexibility in the direction of sending out the signal. We will show the modified infrared transmitter in Chapter 6.

5.3 GLASGOW CONTEXT-SENSITIVE FRAMEWORK

Conventional desktop PCs operate in a very static environment. Most of the interactions between a user and a desktop system are explicit, for example, mouse click/movement, keystroke, etc. Unlike desktop systems, users and devices move around. Computational services highly depend on their

user's physical environments in a mobile computing environment. This type of environment is rich in implicit information regarding the user's current situation. In the case of a mobile computing system, however, explicit input especially keyboards and display from the user may result in a burden due to the limits of hardware design in mobile computing devices. This creates a greater need for mobile computing devices to take advantage of implicit information from their users' current surroundings and environment. This would reduce explicit input required from the users.

The GCS takes advantage of communication between a mobile computing device and sensors, to detect, interpret, and respond to the current situation of their users in the environment. Dey et al. built a set of APIs, the Context Toolkit, for constructing context-sensitive systems (Dey, 2000). These APIs were built in the manner of Graphical User Interface (GUI) components. GUI components can be reused. They also hide the details about specifics of physical interaction between devices from the designers; therefore, the impact would be minimal if the underlying interaction devices are changed. For example, a mouse or touchpad can generate the same events and need not cause any change to the application. This is an important factor for us to consider when deciding how to build sensor components.

The following sections describe the Glasgow Context-Sensitive Framework (GCSF), which is the API we developed to enable the designers to build applications based on the GCS environment. This API, in particular, enables the designer to exploits mobile computing devices' built-in infrared port to communicate with domestic infrared remote controls. Also, two different type of implementing GCS applications are described.

5.3.1 SENSOR API

Normally, the data provided by sensors are not in a form that can be exploited easily by applications. For example, a mobile device's built-in infrared port may report that a particular signal emitted from an infrared transmitter has (probably) been detected whereas an application really wants to know that the user has just approached an object or passed a location. Sensor data must be interpreted to provide a consistent, reliable and abstract view of the status of the current situation, and changes in that status in order to become useful to applications. In the GCS, we developed a sensor API to implement this approach. The sensor API includes a sensor interface, a RawIR sensor implementation, infrared data sets, and utilities that convert raw binary data to hexadecimal values and match the converted data and the pre-cached infrared data set in order to provide reliable sensed data. The matching mechanism is described in section 5.3.1.2.

5.3.1.1 SENSOR INTERFACE AND RAWIR SENSOR IMPLEMENTATION

We began by defining the specification of a set of methods for a sensor. We consider that a sensor consists of mechanisms such as activation, close (deactivation), reading sensed data, and polling frequency. More specifically, we focus on the passive sensing approach that the user's mobile

computing device can detect sensor signals from the immediate physical environment; therefore, the sensor in our specification is either built-in or connected to the mobile computing device through a serial, universal serial bus (USB), parallel interface. We defined that a sensor should be able to be activated and deactivated. The main function of a sensor should be able to read the incoming signals from other transmitters (i.e. infrared transmitters) deployed around the physical environment. The designers can analyse the received data and exploit these as an identifier of location or object. The polling frequency is a flexible interface for the designers to monitor the sensor they create. For instance, an infrared sensor can be monitored less frequently than a GPS sensor. The sensor interface is shown as follows:

```
//*****************************************************************************
//* activate                                                                  *
//*****************************************************************************
        /**
         * turn the sensor on
         *
         */
        activate();
//*****************************************************************************
//* close                                                                     *
//*****************************************************************************
        /**
         * Close the currently functioning sensor
         *
         */
        close();
//*****************************************************************************
//* receiveBytes                                                              *
//*****************************************************************************
        /**
         * Return the sensed data
         *
         */
        receiveSensorData();
//*****************************************************************************
//* PollingRate                                                               *
//*****************************************************************************
        /**
         * Assign the polling rate to monitor the sensor
         *
         */
        pollingRate();
```

A polling approach was used to simulate the thread in Java because there are not supported in Waba. In Java the designers can implement multiple threads which are timed instances that can be used to generate events based on schedule. In Waba the Timer class is used to schedule the control of user

interfaces and their events. We exploit the Timer class to implement the polling mechanism. It acts like a server waiting for data coming from a target. The relevant procedures, for instance, data conversion and notification about changes can be triggered once any change is detected.

The sensor access defines what all sensors must be able to do. They may do more, but this is a minimum. Right now, one way to implement the specification for a raw infrared sensor is as follows (example code in Waba):

```
public class RawIR implements Sensor
{
    boolean activate();
    {
        // do whatever is necessary to activate the RawIR sensor and return callback,
        // TRUE, if it is on successfully.
    }

    boolean close();
    {
       // do whatever is necessary to close the RawIR sensor once it needs to be
       // closed.
    }

    int receiveBytes(byte[] rawIRData, int start, int count)
    {
        // the payload of the byte array is from byte[start]..byte[count]

        // do whatever is necessary to read the sensed raw binary data and store
        // in an array. The content of the array need to be analysed and converted to
        // a proper format. It then can be used as context of location and
        // surroundings information.
    }

    int pollingRate(int millisec)
    {
        // set a polling rate to check whether any change at the sensor.
    }

}
```

The complete implementation of the method `activate` in this example should also include a resource address and operation state of the sensor on a mobile computing device. For instance, the RawIR sensor can be treated as a serial port with an infrared transceiver attached so the resource address is the port number on the device. The operation state includes the communication rate (baud rate) in this case. This method accepts a baud rate and a port number in integer format as its parameters. It returns a Boolean value to indicate whether the activation process has been successful. Similarly, the method close should indicate what should be deactivated. In our GCS, Virtual Notelet

application (fully described in Chapter 6), the polling rate was set to second-scaled. The user's mobile computing device senses the infrared signals from the physical environment discretely. The details about how to analyse the received raw data is described in the next section.

5.3.1.2 RAW SENSED DATA CONVERSION AND IDENTIFICATION

We have opted to use off-the-shelf domestic remote controllers to reduce the costs and complexity of implementing the system. These can be programmed to generate relatively simple numeric patterns that cannot easily be associated with an anchor of information, such as a URL. In consequence, we rely on a "lookup table" that associates information about an object or a place with each of the different patterns programmed into the remote controls. It could also use combinations of differential signals and infrared transmitters as well. When the users' device detects a new infrared signal it uses the table to look up associated information. Each button on the domestic infrared remote control can fire different signal patterns. Our approach is to analyze these signals to elicit consistent data patterns. The data patterns are then recorded in hexadecimal format and can be assigned to represent an object or place. The reason for converting the sensed infrared data from binary to hexadecimal format is that we want to minimize the identification search cycle. As we noted previously, a domestic infrared remote control sends signals repeatedly until the button on the remote control is released. Unfortunately, it was discovered that the same button can yield seemingly anomalous results. This causes difficulties when exploiting the signals as identifiers for physical entities. It is because alternative RawIR polling mechanisms cannot predict the point in the remote controller's output signals cycle when it was intercepted. In addition, the infrared signals fired by pressing the same button on the remote can change one or two bytes in the data stream. This complicates the identification. In such cases, a tolerance in the comparison could be used, but the risk of confusion between different signal streams would be increased if the tolerance is set too high. Our solution was to find out the repeats in the signal data streams that represent each button on the remote control from the IRTest displayed result. We pre-cached the repeats as data patterns in the IRData, a part of the API, and implement a search, which traverses the input data stream and calculates the proximity to the possible identification. To formalize the search problem we adopt the approach proposed by Nievergelt et al described as follows (Nievergelt and Hinrichs, 1993):

*"**Problem:** Given a (long) string $r = r_1 r_2 \ldots r_n$ of n characters, where r = received raw infrared data and a (usually much shorter) string $c = c_1 c_2 \ldots c_m$ of m characters, where c = pre-cached infrared data pattern (the pattern we are looking for), find all (nonoverlaping) occurrences of c in r. By sliding a window of length m from left to right along c and examining most characters r_i m times we solve the problem using $m \cdot n$ comparisons. By constructing a finite-state machine from the pattern c it suffices to examine each character r_i exactly once. Each state corresponds to a prefix of the pattern, starting with the empty prefix and ending with the complete pattern. The input symbols are the input characters $r_1 r_2 \ldots r_n$ of r. In the j-th step the input character r_j leads from a state corresponding to the prefix $c_1 c_2 \ldots c_i$ to:"*

- *The state with prefix $c_1 c_2 \ldots c_i c_{i+1}$ if $r_j = c_{i+1}$*

- *A different state if $r_j \neq c_{i+1}$*

The following shows the description of the search we implemented based on (Nievergelt and Hinrichs, 1993):

1. Compare the first element in the received raw infrared data array with the first element in the pre-cached infrared data array.

    - If both of the elements are not equal, then compare the first element in the received raw infrared data array with the following element in the pre-cached infrared data array till match found.

    - If the match cannot be found in the current row of pre-cached data set, then move to the next row of the data set and restart this step.

    - If the first element in the received infrared data array finds matched element in the pre-cached data array then:

2. Compare the elements next to the matched ones in both the received raw infrared data array and the pre-cached data array.

    - If both of the elements are not equal, then compare the next elements in both the pre-cached data array and the received raw infrared data array. If at the end of the pre-cached infrared data array, the next element in the array is the first element and so forth.

    - If the elements are equal, match count increases.

3. Recursively apply step 2 to the arrays till the end of the received infrared data array is reached. The identification can be found once the match count reaches the tolerance value.

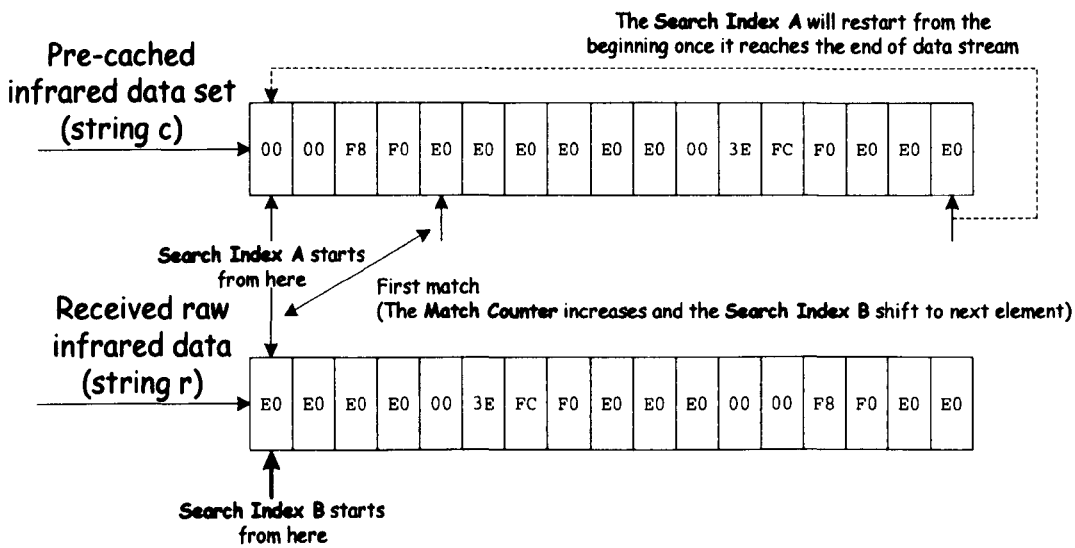


Figure 5.5 Infrared Data Identification Search

The sensors used in most context-sensitive mobile computing systems are different. Building a sensor module, which acts in a way similar to a GUI component, would ease the difficulties in building such a system. Our approach is to wrap the RawIR sensor so that it resembles the API to be a GUI component. This is done by extending the Waba Control class. The Control class is the ancestor of GUI classes such as Button, Checkbox, Label, and etc. It contains methods for event handling and the "painting" of controls. An alternative to this approach is to incorporate the Timer into the RawIR control. As mentioned, the RawIR sensor adopts a polling mechanism to monitor the raw infrared port. In Waba the "Timers" are associated with user interface controls. Any user interface control in a Waba program can have its own timer. When a timer ticks, it generates a "`ControlEvent.TIMER`" event which is sent to the user interface control associated with the timer. We associated a Timer with the RawIR sensor control so it activates and closes the raw infrared port iteratively with a specific frequency. Unlike conventional GUI components, which provide the visible portion of a programme, the RawIR sensor control is invisible. However, similar to common GUI components, we need to assign an event type for the RawIR sensor control. Normally, there are three steps, which the GUI component event handler would perform. Firstly, identify the source of the event. Secondly, identify the event type. Thirdly, invoke the appropriate routines to execute the relevant procedure based upon the event. In the case of RawIR sensor control, once the raw infrared port detects and receives signals, the RawIR sensor control simply fires the equivalent of a mouse click or touch screen pressed down event. In the meantime, an event handling procedure is started. In Waba the `onEvent` method is where the event handling takes place. The following Waba example code demonstrates how the RawIR control is implemented.

```
public class RawIRControl extends Control
{
     Timer timer;

     RawIR rawIRPort;

     IRDATA IRData;

     public void RawIRControl()
     {
       if (timer == null)      {timer = addTimer(polling frequency);}
     }

     public void polling()
     {
       HexStringFactory(); // do raw data conversion

       buttonSearch(); // do identification search

       postEvent(ControlEvent(ControlEvent.PRESSED));
       // generate event once the incoming signal is matched.
     }
```

```
    public void onEvent(Event event)
    {
      if (event.type == ControlEvent.TIMER) {polling();}
    }
}
```

The following example shows a simple Waba code that would recognize the sensed infrared signals as equivalent to a button pressed down event. The following code demonstrates the implementation in Waba:

```
public class ContextSensitiveTest extends MainWindow
{

    RawIRControl rawIRControl;

    public ContextSensitiveTest()
    {
      rawIRControl = new RawIRControl();

      add(rawIRControl);
    }

    public void onEvent(Event event)
    {
      if (event.type == ControlEvent.PRESSED)
      {
        if (event.target == rawIRControl)
        {
          // do event handling procedure
        }
    }
}
```

The ContextSensitiveTest class extends MainWindow, which every Waba application that has a user interface must extend from. Next we declare the variable, rawIRControl. In the constructor we call RawIRControl to initiate the RawIR sensor component. Notice that we do not layout the placement of the controls.

The function of RawIR sensor control is to poll the raw infrared port and wait for incoming signals. Once signals arrive at the port, it simply identifies the signals and then translates to a hexadecimal value. The hexadecimal values are passed to the identification search module to see whether they are matched to any pre-cached infrared data. The output is an identifier of a button on the remote control. Figure 5.6 illustrates the data flow diagram of the RawIR sensor control.

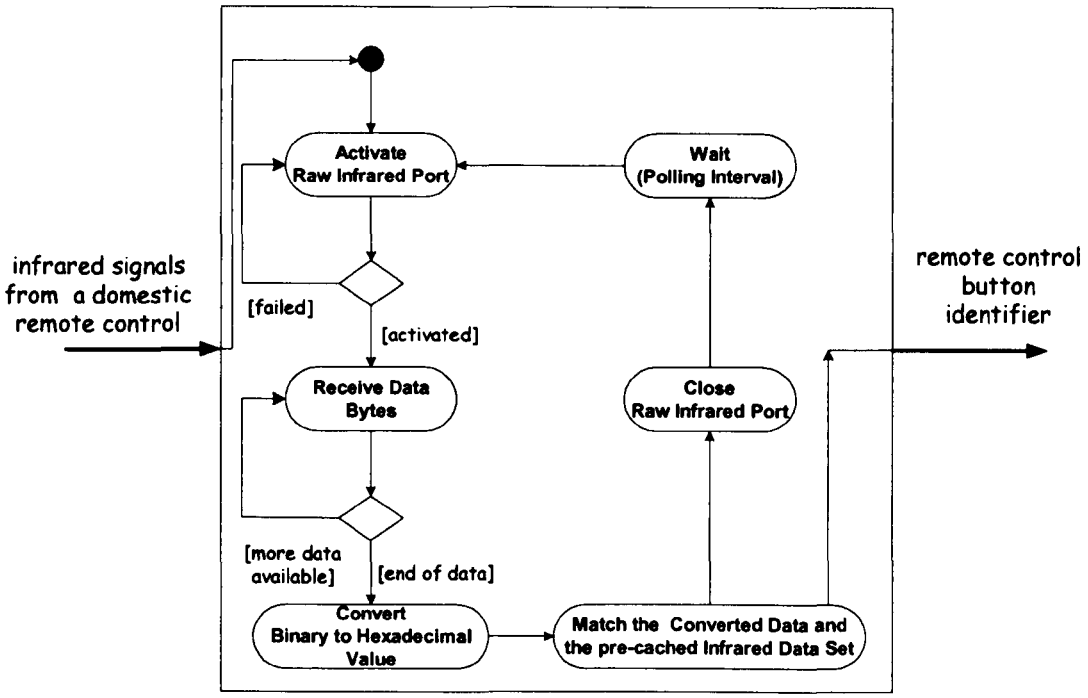


Figure 5.6 RawIR Sensor Data Flow Diagram.

5.3.2 METHODS FOR DEVELOPING CONTEXT-SENSITIVE MOBILE COMPUTING APPLICATIONS USING GCSF

As we noted in Chapter 4, the GCS can operate without mobile telecommunications, such as a 802.11 connection. Location and object dependent information is cached on the mobile computing device. As the user moves around the building in which the system is hosted and infrared signals are detected. Applications can use this data to tailor their presentation of information. A guide map is a good example to demonstrate this approach using the GCSF. The context-sensitive guide map interests researchers in the field of context-sensitive mobile computing (Davies et al., 1999, Long et al., 1996, Oppermann and Specht, 1999, Pascoe, 1997). The consensus on the type of application is that the major context information is the location of the users. The purpose of this application is to show users their current location relative to their environment and provide other details relevant to that location. We developed a navigational map application using static images. The domestic infrared remote controllers were used as location identifiers and were deployed around the building. As the user moves throughout the building, the display shows a "You are here!" on the map. This is purely a demonstration application. As we shall see, however, it has been extended to inform customers of their location with a large warehouse store. This enables them to locate their location relative to products in the store.

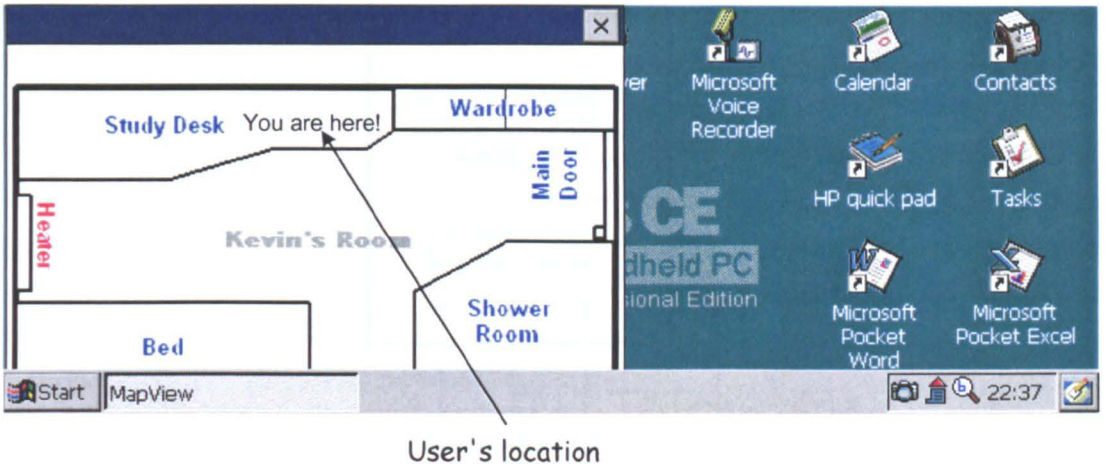


Figure 5.7 HouseView Screen Shot

This example illustrates the simplest form of the GCS. To build such an application designers do not have to worry about sensor details. The GCSF provides the designers with a GUI-like sensor component. It can be added to an application as part of its user interface. This eases the task of context (i.e. location information) acquisition. When the sensor is changed, it requires a minor modification to the application. The interface designer needs to instantiate another type of sensor control on the application. This poses a number of interesting issues regarding the modification. The designer who builds context-sensitive mobile computing applications using the GCSF would face two situations: First, to build their applications using the GCSF without adding or modifying underlying components. Second, the relevant component, for example, the sensor control module for building their application, lacks support in the GCSF. As noted in Dey in Chapter 3 (Dey, 2000), this designer can be categorized as a library and toolkit designer. The former simply exploits the existing support and architectural widgets and later adds or modifies the existing support to suit the specific need. The second situation happens rather often when building a context-sensitive mobile computing system. This is due to the variety of context information that can be sensed/measured and makes it difficult to provide every possible context sensing and interpreting components for all library designers. It also reflects the reasons why we provided a sensor interface to the GCSF users so that they can implement the sensor they need. The rest of the usage of the GCSF would continue in the same fashion once the new sensor is implemented. For instance, the infrared transmitter we used in the demonstration can be substituted for a GPS receiver in an outdoor environment. The toolkit designer can implement a GPS sensor API and a GPS sensor control in a similar manner to the RawIR sensor. The polling rate may need to be adjusted to reflect the different nature of each sensor. As an example, a GPS sensor (receiver) goes almost continuously but infrared sensing is discrete. A context-sensitive mobile computing shopping assistance application, MapView, based on this approach will be described in the Chapter 7.

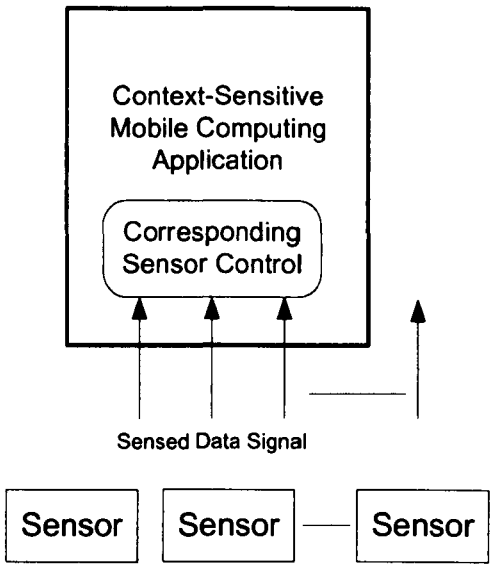


Figure 5.8 The GCS without Remote Server Support.

We exploit web technology to support the development of context-sensitive mobile computing systems. The idea is based upon a "lookup table" that associates a URL with each of the different patterns programmed into the remote controls. When the users' mobile computing device detects a new infrared signal it uses the table to look up an associated URL. Our approach is to trigger a connection with a URL on the Internet once the mobile computing device detects and identifies the signal pattern. The infrared identifier is encoded in HTML request format it looks like: "`http://contextserver/servlet/contextsensitiveApp?IRID="infrared ID"` and this is sent to the context server. The server programme parses the http request, obtains the infrared identifier, uses the identifier to search the look up table, and responds with a web page according to the associated URL or query result based upon the location from the database. The server-side application is implemented in a Java servlet hosted on a Web server. A servlet can be responsible for taking data embedded in HTML format from web-based clients and applying the core function (i.e. business logic) used to update a corresponding database (Bloch and Bodoff, 2001). The underlying concept of this approach is to separate a context-sensitive mobile computing application into two parts, the GUI (plus sensor control) on the client side and the core computation on server side. This reflects the apparent fact that most mobile computing devices provide limited hardware and software resources and stationary machines such as servers are powerful and rich in computational resources. This approach also provides the designers with a clear cut division in terms of design and implementation labour. It can simplify the effort on the client side and at the same time leverage off the power on the server side.

In brief, the client side implementation in the approach needs to complete the tasks: connect to the context server, send detected infrared identifier, receive the output from the server, display results, and close the connection. The server receives the request from the client, process it, and send out the result

to the client. This is similar to the Model-View-Controller (MVC) design pattern. It is a software architectural pattern approach, which divides an interactive application into three components: model, view, and controller. The model component contains the core function of an application and deals with the application data. The application output is channelled to the view component, which displays the information to the user. The controller deals with the user input and acts as a middleman between the model and view component. A user interface consists of a view together with a controller component (Buschmann, 1996, Sage, 2001, Schigeoka, 2002). In this approach, the sensor control can be considered as a controller component and the information display area in the client programme can be regarded as a view component. The model component, which contains the core function and data of the application, is implemented in the server. Interface separation has raised many problems and further experience in the development of mobile systems is needed to see if we can preserve this possible separation.

A web-based annotation system for physical objects, which is a demonstration of this development approach, will be described in Chapter 6.

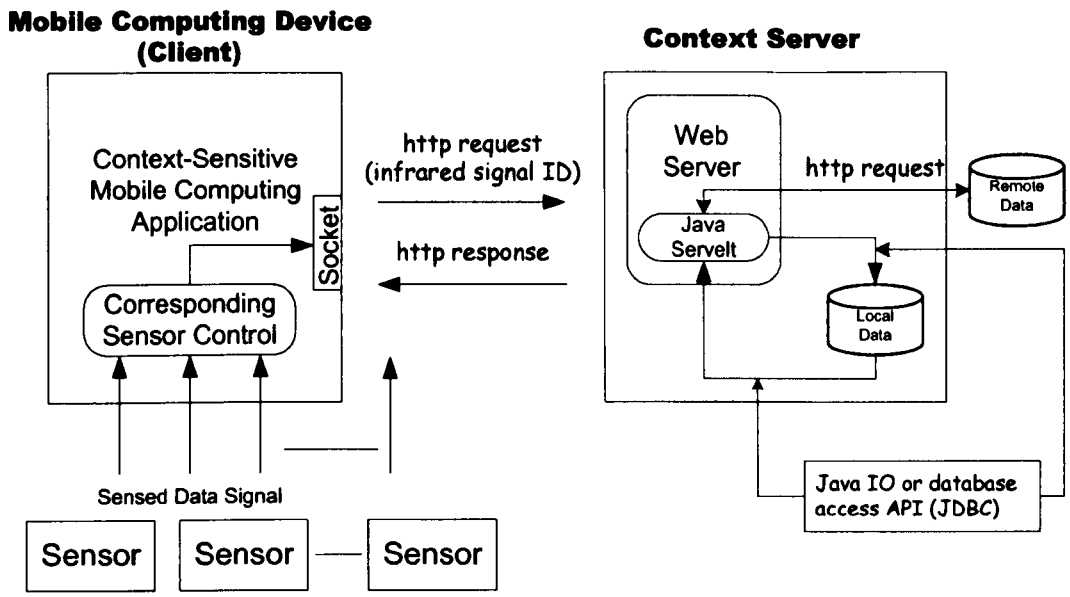


Figure 5.9 The GCS with Web Server Support.

5.4 SUMMARY

This chapter has focused on the technical challenges during the implementation of GCS and has detailed the solutions for them. The key challenges are:

- Select a programming language that can suit the requirements of implementing the GCS.

- Tackle the issue of differences in communication protocols and behaviours between mobile devices and domestic infrared remote controllers.

In this chapter, the implementation of the API enabling designers to build GCS applications has also been described. Two demonstrations of the use of the API to build the GCS applications are presented. The GCS is designed to provide a prototyping environment for context-sensitive mobile computing systems based on off-the-shelf building blocks. The off-the-shelf approach was born from pragmatic constraints. The development of context-sensitive mobile computing systems requires considerable engineering skills. This could be an obstacle to HCI designers who want to discover more about detailed social or usability issues but do not possess knowledge about the hardware or software design and implementation for this type of systems. In this case, the GCS can only provide the HCI designers a preliminary idea of how the prototyping environment works and what building blocks might be needed. They, however, have to cooperate with software developers in terms of the GCS application development. As for the development of the applications based on the GCS API, the users (software developers) of this API can be categorized into library and toolkit programmers. The former simply exploit the existing support whereas the latter will alter and add APIs that have not existed to suit their needs in the application development. In the case of change of underlying hardware components, such as sensors, the off-the-shelf approach can help minimize cost and impact to the software developers. For example, it is possible to update system components when new technologies, such as Bluetooth, become widely available. They, toolkit programmers, can simply implement corresponding sensor controls to access and interpret the sensor data and make them available to the applications.

In the next two chapters, we will describe the design and implementation of the GCS applications. The design of these applications is based on the design process described in Appendix B. It revisits the task analysis technique we used in Chapter 2. This helps us to identify context when we developed applications in Chapter 6 and Chapter 7.

CHAPTER 6
CASE STUDY 1: THE VIRTUAL NOTELET APPLICATION

The use of paper-based annotations such as Post-It notes is very common in our daily life (Baldonado et al., 2000, Coschurba et al., 2001, Pascoe, 1997). An annotation is a text or drawing with numerous uses. Baldonado et al argued that an annotation is a commentary on an object. They also comment upon the role of the users in using annotations (Baldonado et al., 2000). The annotation is not limited in paper-based form and the meaning of the annotation can be extended to a commentary on physical entities such as a place, surrounding objects, etc. We noticed that many staff in our department attach information related to their job (i.e. research topic, teaching, or administration) activities, personal interests, in/out status, etc. on their office door. Figure 6.1 provides an example of the information that is stuck on an office door.

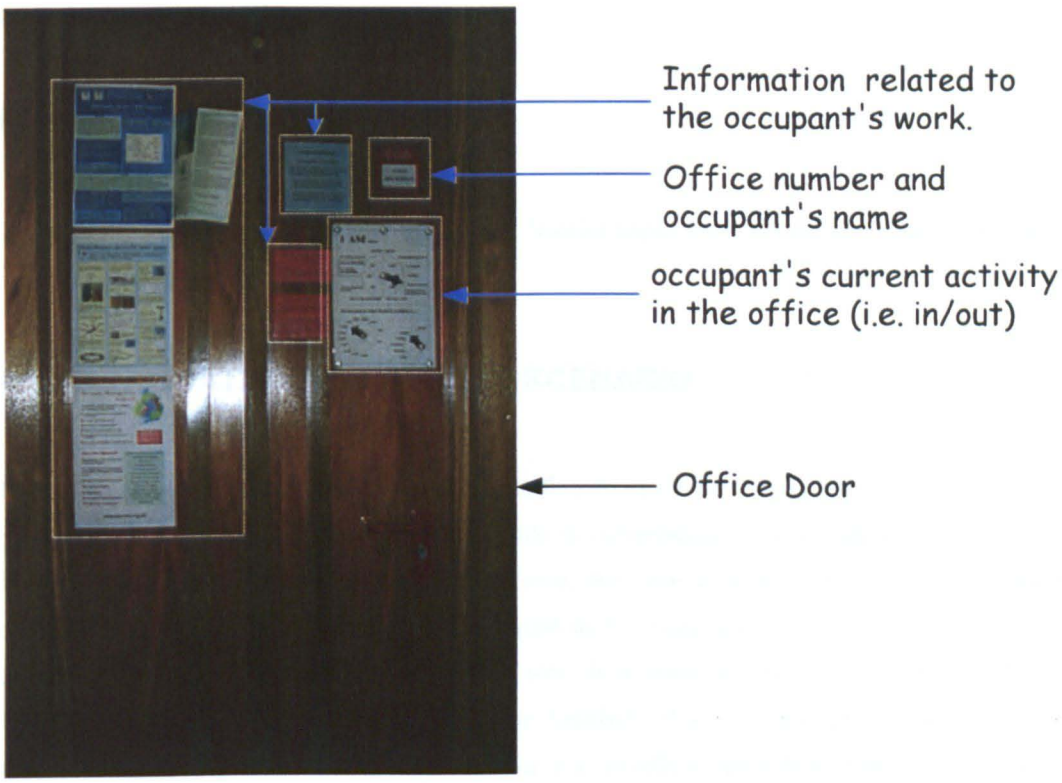


Figure 6.1 Typical Information Attached on an Office Door.

As mentioned, the GCS environment has been developed to enable interface designers to construct context-sensitive mobile computing applications using off-the-shelf techniques. In order to demonstrate the GCS, we focused development work on a proposal made by previous researchers (Burrell and Gay, 2002, Espinoza et al., 2001, Pascoe, 1997, Selker, 2000, Spohrer, 1999). The idea is

to implement an application that enables its users to stick virtual notes to physical objects, such as doors. These annotations can then be read by holding a PDA close to the object. Alternatively, users might read the notes on their door from any location that offered Internet connectivity, following the model proposed by the cooltown project. This, Virtual Notelet application, was chosen because a number of previous researchers had claimed significant benefits from such systems. In particular, the Virtual Notelet system is based upon the "context board" attached outside many office doors. A context board is used to display a user's status in their office and also allows a user to specify messages through a predefined template or specific selection when leaving their office. For instance, when a user is leaving her office she can leave messages on the door by adjusting the indicator on a context board or attaching paper based post-it notes for further information. The user can view any message left by others in her absence along with the time left when she returns to her office.

The Virtual Notelet presents a notion of utilizing wireless-enabled mobile computing devices as an interface to bridge the gap between physical and digital space. It provides the user to be able to have the feeling of touch and manipulates the digital information virtually on physical objects. This reflects the argument, which the GCS operating environment is similar to Augmented Reality, mentioned previously.

6.1 APPLICATION DESIGN

The following sections discuss how the Virtual Notelet application is built using the design process described in Appendix B.

6.1.1 APPLICATION DOMAIN AND SCENARIO

Interaction at office doors happens frequently. Office doors do not simply act as physical barriers to particular rooms. They also play a significant role in communicating information about the location and availability of the occupant. In a wider sense, they can also be thought of as a medium of communication for information from the occupant to her colleagues and vice versa. The doors to communal and shared locations play a similar role. It is often possible to tell if a room has been booked by looking for notes attached to the door. Similarly, if a meeting is being conducted then the same approach can be utilized to indicate whether it is socially acceptable to interrupt that meeting or not. People in the office domain may walk by an office door in order to find out if a colleague is in her office. In this situation, the visitor encounters a problem about whether to enter or not whereas the occupant faces intrusion. How does the visitor put an anchor to the uncompleted activity in order to resume the interaction with the occupant if the occupant is not in the office or unavailable for the visitor? From our preliminary observation, some staff in our department use annotations to indicate their current status in the offices. Some of them provide their visitors with Post-It notes, which are attached on the door, so they can leave messages on the door. Some authors argued that people's

actions at a door are determined by the status of the occupant in the office (Selker, 2000). Their observation shows most visitors perform the action "knocking and waiting", "checking status", or leaving notes". The actions such as "walking in" and "knock and walking in" are rarely happen. This reflects the importance of the annotation on an office door.

There are two situations we are interested in. Firstly, a person stands in front of an office door and checks the status of the occupant. Secondly, an office occupant stands in front of her office door, and manipulates the annotations to indicate her in/out status, check the messages left by the visitors, and opens the door in order to enter her office or close the door to leave her office. In the first situation, we think the person is willing to visit the occupant and interested in her status in the office. According to the design process mentioned previously, the context in this situation is "a person stands in front of an office door". We assume the context means the person is willing to visit the occupant in the office. However, there may be a variety of other scenario. In the second situation, the context is "an office occupant stands outside in front of her office." When this happens, the occupant is going to enter or is just about to leave her office. Both of the situations infer the subsequent actions. When the first situation happens the person checks the status of the occupant in the office and decides whether to enter or leave messages on the door. The office occupant has to adjust her in/out status and check messages left on the door by others when the second situation happens. The situations in the scenario of a user standing in front of a conference room are similar to the scenario of the office door. The user may be the presenter, an invited or an exclusive audience. The same approach can be used to determine whether the user can enter the conference room or not.

The Digital Threshold project, which is based on the office scenario, uses two switches (flat touch-sensitive screens) installed on both the inside and the outside of a door. The switches can notify whether the office occupant has an appointment, identify people, allow the office occupant to respond and update the occupant's calendar (Selker, 2000). Visitors can see the occupant's schedule and reschedule a meeting or leave messages. It provides both occupant and visitor with the ability to scheduling meetings on the fly. However, the occupant has no privacy control.

6.1.2 LIGHTWEIGHT USER MODEL

A user model describes a user's state and preferences. In the scenario of interaction at a door in the office domain, the user model needs to be extended to include the social aspects in the interaction. In the previous section, we indicated two different roles, visitor and occupant, in the scenario. Social barriers can be intuitively described in the scenario. For example, the visitor are not supposed to open the occupant's office door without permission, and the visitor cannot adjust the occupant's in/out status and remove notes left by others attached on the door. Recognising the roles in a scenario can be very important in terms of shifting user tasks to the computer tasks (Dix et al., 1998, Faulkner, 1998). User related information in the Virtual Notelet includes the user name, office number, and most recent location. This information allows the Virtual Notelet to be aware of the role of the user and their

location and then applies the access control rule on virtual notes to them. This raises interesting issues: can we remove physical/virtual notes left by other people? Who has the right to remove these notes?

6.1.3 TASK ANALYSIS

As we mentioned in Chapter 2, the context information for an application can be considered as the inputs that are required to perform the tasks that the application intends to perform to help its user achieve a certain goal in a situation. The input mentioned here is referred to implicit input that is not intentionally performed by the user to operate a system but conceived as input by the system (Schmidt, 2000). For instance, a user carrying a handheld computer is interested in the artifact in front of her in a museum resulting in the information about the artifact displayed on the computer. The user performs the task, "walk to the artifact interests her", is regarded as an input to the system to perform system tasks to suit the user's current situation.

The user and computer task models in the Virtual Notelet is that the users carries wireless-enabled mobile computing devices and walks around in the building. When the user stands in front of an office door, the computer task is to display the occupant's status in the office and virtual notes attached on the office door. The user task is that the user, who is the occupant of offices, can adjust her status in their office. Others can decide whether to knock the door or leave virtual notes depending on the occupant's status in the office.

The following describes three different situations when using the Virtual Notelet. The user task at office door is described using HTA to show a high level view of the interaction. We also utilize the entity-relationship-based approach to figure out the entities, actions, actors involve in the interaction within the scenario. Also, the relationship between the entities, actions, and actors is important when we try to transform user tasks to computer tasks.

Situation 1: Approaching a colleague's office and want to know her status in the office when standing in front of the door.

High level view of context: a visitor is approaching and then standing at an office door.

Role: visitor

```
0. in order to meet the colleague in her office
      1. walk by the office door
      2. check the context board attached on the door
      3. leave a note (using  Post-It notes)
            3.1 write message on the note
            3.2 detach the note from the pile of Post-It notes
            3.3 attach the note on the door
```

4. knock the door
5. wait few seconds
6. open the door
7. walk in

Plan 0: do 1-2-4-5-6-7 in that order
When the occupant is away from the office or busy in the office do 3

Plan 3: do 3.1-3.2-3.3 in that order

Object Visitor **human actor**

Actions:

V1-1: walk to the office
V1-2: check occupant's status showing on the door
V1-3: leave notes
V1-4: knock, open the door, and walk in the office

Object Post-It note **simple**

Attributes:

Affordances: hold/fold/attach/detach/draw or write

Events:

E1-1: occupant is free in the office
E1-2: occupant is busy in the office
E1-3: occupant is not in the office

Relations: object-object

Location (Post-It notes, office door)
Location (context board, office door)

Relations: action-object

patient (V1-2, context board and notes)
- Visitor "sees" the context board and notes attached on the door

instrument (V1-3, Post-It notes)
- Visitor writes down messages on the note and attaches it on the door using its self-attaching area on the back

patient (V1-4, door)
- Visitor knocks and opens the door

Relations: action-event

before (V1-2, V1-3)
- Visitor must check the office occupant's status before deciding whether to leave a note

triggers (E1-1, V1-4)

- "Occupant's status is free" triggers the visitor to knock, open the door, and walk in the office

triggers (E1-2, V1-3)

- "Occupant's status is busy" triggers the visitor to decide to leave a note

triggers (E1-3, V1-3)

- "Occupant's status is away" triggers the visitor to decide to leave a note

As mentioned previously, the task analysis is not an end in itself. For instance, the events listed in the previous paragraph are very unlikely to provide a complete description of the changes that must be considered by the system. In contrast, the HTA represents an initial stepping stone between the informal scenario and the more detailed information required to move towards a prototype implementation. Both the scenario and the task analysis are refined by the insights that are provided once users can access the system. Considering building a context-sensitive mobile computing application to help the user perform these actions we should determine what actions the application needs to perform and what input it expects. From the task analysis listed above, there are two human actor actions, V1-2 and V1-3, we are interested in. In more detail, the application should display the occupant's status in the office and provide a Post-It note like function so that a visitor can write messages and post it on the office door. From the application point of view, when its user stands at an office door it must first identify his/her role in the ongoing interaction. This can be done by requiring the user to "login" so the application knows whether s/he is a visitor or an occupant in the office. The login process can be implicitly adapting sensing technology such as iButton mentioned in previous chapters or explicitly asking the user to type in his/her ID and password. Once the user's role is obtained, the application can perform subsequent actions. For instance, the application displays the occupant's status in the office. The acquisition of the information is described in situation 2. As shown in the action-event relations section, the occupant's status in the office determines the visitor's subsequent action, "leave a note or knock the door". The occupant's status can be regarded as an input to the application to activate its Post-It like function to the user. The user can write messages on the virtual note and attach it on the office door.

Object Virtual Notelet **non-human actor**

Actions:

VN1-1: identify the user's role

VN1-2: display the occupant's status in the office

VN1-3: activate note editor

VN1-4: associate the virtual note with the physical office door

Object Virtual Post-It note **simple**

Attributes:

Affordances: virtually attach/detach/draw or write

Relations: object-object

Location (virtual note, computing device (i.e. PDA))

As we mentioned in Appendix B, this design process help us to extract a high level view of the context information in a scenario. From the application's perspective, designers of context-sensitive mobile computing applications must decompose the context information into the required inputs to each application. We are using an existing form of task analysis from (Dix et al., 1998) instead of developing our own approach. The following shows the context information, which includes both explicit and implicit inputs, required by the application that implements situation 1.

Input to Virtual Notelet

user ID and password → VN1-1
occupant's status in the office (i.e. in/out) → VN1-2
occupant's "bust" or "away" status → VN1-3
user fires "attach" command → VN1-4

Context in Virtual Notelet

[user] → VN1-1
[office occupant's status] → VN1-2, VN1-3
[virtual note manipulation command] → VN1-4

The following describes the occupant's interaction at her office door when coming back to the office.

Situation 2: Office occupant is approaching to her office door from outside of the office and she wants to adjust her in/out status and check the notes left by others when she stands in front of the door.

High level view of context: an office occupant approaches to her office and stands in front of the door when arrives.

Role: occupant

0. enter her office
    1. walk by her office door
    2. adjust the in/out status on the context board on the door
    3. check the notes attached by visitors on the door
    4. remove notes
    5. open the door
    6. walk in

Plan 0: do 1-2-3-5-6 in that order
    if any note attached on the door then do 4

Object Office occupant **human actor**

Actions:

V2-1: walk to the office

V2-2: adjust in/out status showing on the door

V2-3: check and remove notes from the door

V2-4: open the door and walk in the office

Object Post-It note **simple**

Attributes:

Affordances: hold/fold/attach/detach/draw or write

Object context board **simple**

Attributes:

Affordances: adjustable indicator

Events:

E2-1: notes left by the occupant are attached on the door

E2-2: notes left by other visitors are attached on the door

Relations: object-object

Location (Post-It notes, office door)

Location (context board, office door)

Relations: action-object

patient (V2-2, context board)

- occupant adjust the in/out status displayed on a context board

patient (V2-3, Post-It notes)

- occupant "see" and remove the notes attached on the door

patient (V2-4, door)

- occupant open the door and walk in

Relations: action-event

before (V2-2 or V2-3, V2-4)

- occupant adjust her in/out/busy/free status and check notes attached on the door before she enter the office

triggers (E2-1 or E2-2, V2-3)

- notes left and attached on the door by the occupant or others trigger the occupant perform the action "remove the notes"

When the user arrives at her office door from outside the application must identify the relationship between the user and the office as described in situation 1. If the user's role is identified as the occupant of the office she can adjust her in/out status manually on the computing device or implicitly updated by the application if it embodies a more sophisticated user model (i.e. meeting schedule, location, and etc.). If a virtual note has been left by others or the occupant herself, the application displays the notes on the user's computing device.

Object Virtual Notelet **non-human actor**

Action:

VN2-1: identify the user's role
VN2-2: activate the virtual context board
VN2-3: display virtual notes and provide the user with note manipulation function
VN2-4: modify the relations between the virtual note and the physical door

Object virtual context board **simple**

Attributes:

Affordances: virtually adjustable indicator

Object Virtual Post-It note **simple**

Attributes:

Affordances: virtually attach/detach/draw or write

Relations: object-object

Location (virtual note, computing device (i.e. PDA))
Location (context board, computing device (i.e. PDA))

Input to Virtual Notelet

user ID and password → VN2-1
occupant stands at the door → VN2-2
notes attached on the door → VN2-3
user fires "remove" command → VN4

Context in Virtual Notelet

[user] → VN2-1
[occupant's location] → VN2-2
[virtual note attached on the door] → VN2-3, VN2-4

The following describes the occupant's interaction at the door when leaving the office.

Situation 3: Office occupant is approaching to her office door, opening, walking out, and locking the door. She adjusts the in/out status and may leave notes to state further information when she stands at the door.

Role: occupant

```
0. leaving her office
        1. open the door
        2. close the door
        3. adjust the in/out status on the context board attached on the door
        4. check and remove notes
        5. leave a note for extra message
                5.1 write message on the note
                5.2 detach the note from the pile of Post-It notes
                5.3 attach the note on the door
        6. lock the door and leave

Plan 0: do 1-2-3-6 in that order
         if any note attached on the door then do 4.
         if further message is needed then do 5

Plan 5: do 5.1-5.2-5.3 in that order
```

Object Office occupant **human actor**

Actions:

V3-1: walk to the office door
V3-2: open the door, walk out, and close the door
V3-3: check and remove notes from the door
V3-4: adjust in/out status showing on the door
V3-5: lock the door

Object Post-It note **simple**

Attributes:

Affordances: hold/fold/attach/detach/draw or write

Object context board **simple**

Attributes:

Affordances: adjustable indicator

Events:

E3-1: occupant is free in the office
E3-2: occupant is busy in the office

Relations: object-object

```
Location (Post-It notes, office door)
Location (context board, office door)
```

Relations: action-object

```
patient (V3-2 and V3-5, door)
        - occupant open, close, and lock the door
patient (V3-3, Post-It notes)
        - occupant check and remove the notes
patient (V3-4, context board)
        - occupant adjust her in/out status displayed on the context
        board
```

Relations: action-event

```
trigger (E3-2, V3-3)
```

When the user stands at the office door, the application must identify the relationship between the user and the office as described in situation 1 and 2. If the user's role is identified as the occupant of the office she can adjust her in/out status manually on the computing device or it can be implicitly updated by the application if it embodies a more sophisticated user model (i.e. meeting schedule, location, and etc.). If a virtual note has been left by others or the occupant herself, the application displays the notes on the user's computing device. From the user task analysis listed above, we see that the office occupant's status and notes that attached on the office door interest both a visitor and an office occupant when they stand at the door and influence their subsequent tasks.

Object Virtual Notelet **non-human actor**

Action:

```
VN3-1: identify the user's role
VN3-2: activate the virtual context board
VN3-3: display virtual notes
VN3-4: modify the relations (remove) between the virtual note
and the physical door and provide the user with note
manipulation function
```

Object virtual context board **simple**

Attributes:

Affordances: `virtually adjustable indicator`

Object Virtual Post-It note **simple**

Attributes:

Affordances: `virtually attach/detach/draw or write`

Relations: object-object

```
Location (virtual note, computing device (i.e. PDA))
Location (context board, computing device (i.e. PDA))
```

Input to Virtual Notelet

```
user ID and password → VN3-1
occupant stands at the door → VN3-2
notes attached on the door → VN3-3
user fires "remove" and "create" virtual note command → VN3-4
```

Context in Virtual Notelet

```
[user] → VN3-1
[occupant's location] → VN3-2
[virtual note attached on the door] → VN3-3
[occupant's status in the office] → VN3-4
```

To sum up, the context information supported by the initial prototype of the Virtual Notelet application will include "the role of the user", "the user's location", "office occupant's status", and "virtual note on the office door". The user's location triggers the information presentation about the occupant's status in the office and notes left by the occupant or others attached on the door. The context, occupant's status in the office, determines the visitor's interaction at the office door such as leaves a note or "knocks and walks in".

6.1.4 VIRTUAL NOTELET OPERATING ENVIRONMENT

The Virtual Notelet prototype was implemented and deployed in the Computing Science Department, University of Glasgow. Users carry wireless-enabled PDAs while they are walking around within the department. We chose this location for the deployment of an initial prototype because we were familiar with the context of use and could also perform initial debugging of the GCS environment before giving the system to its eventual users in the department. The PDA can receive a location identifier from infrared transmitters mounted on the doors and communicates the location identifier and the user's identifier (implicit input) back to the context server over a radio frequency LAN. Users can get the information about the office from the context server on the PDA (actions). The context of an office is, for example, in/out, at meeting progress, go to somewhere and will be back at what time. If the information shows the occupant is not in the office or cannot be disturbed due to some reasons, visitor can leave and attach virtual notes on the office door using the PDA (actions). The Virtual Notelet will be displayed on the occupant's PDA (actions) when she comes back to her office and stand in front of the door (context) within the coverage of infrared transmitter.

6.1.5 ACTIONS IN THE VIRTUAL NOTELET

The Virtual Notelet exploits the context information mentioned in section 7.1.3 to implement and perform context-sensitive actions. Figure 7.2 illustrates the data flow diagram of the application. When the user's PDA detects the infrared signal, the application communicates the user's ID, office number, and infrared identifier back to the server. The information can be used to identify the user's role in the interaction at the door, where the user is currently standing at, and inform the subsequent appropriate interaction for the user at the door. The user's role is defined to be [occupant, visitor]. The intention was that other roles might be added in response to observation from the initial evaluation. These are the roles that were highlighted during the task analysis described in the previous sections. The components of the Virtual Notelet application are categorized as follows:

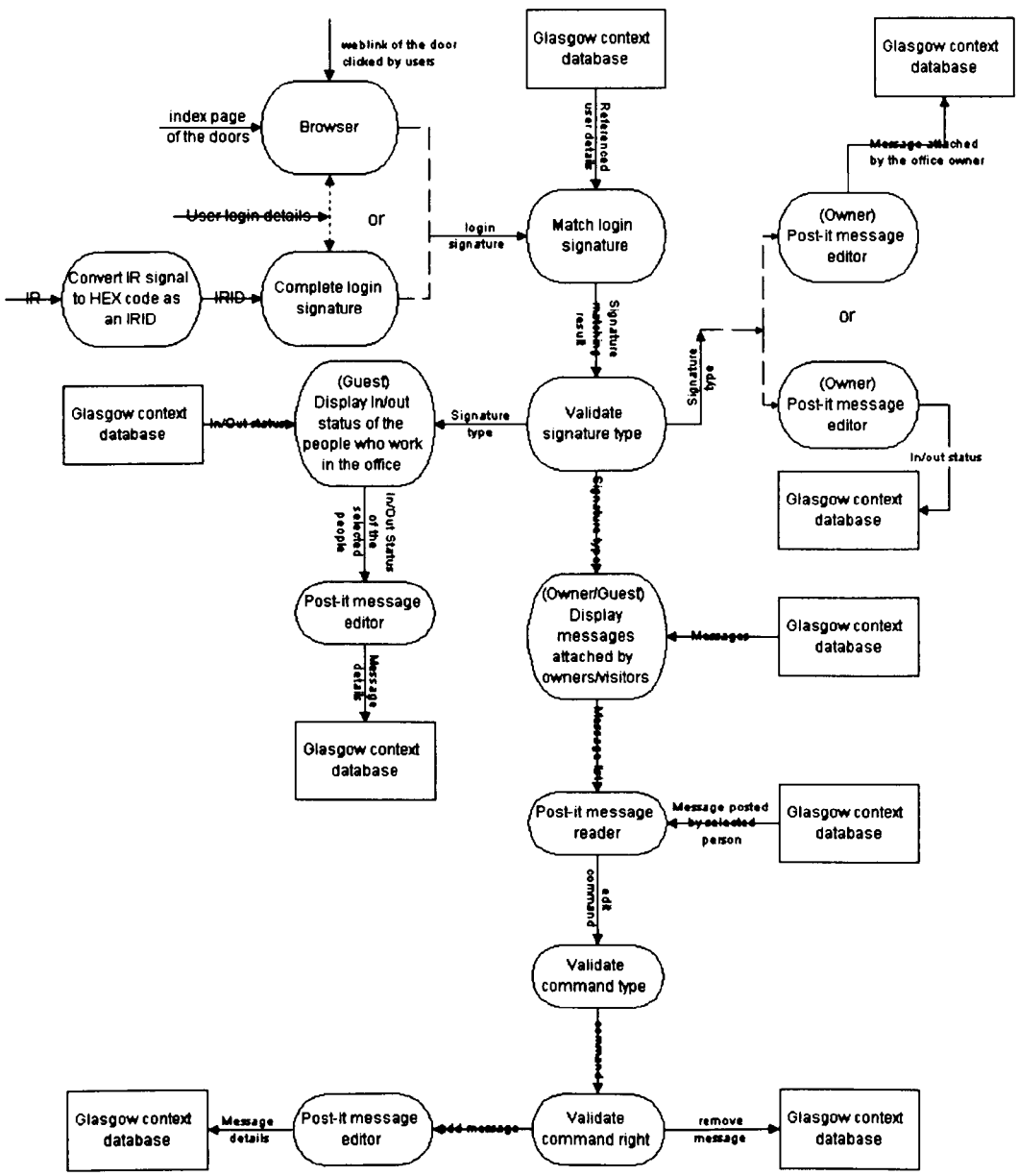


Figure 7.2 The Virtual Notelet Data Flow Diagram.

Server

In order to implement the Virtual Notelet prototype we had to specify an architecture that is able to associate information with physical entities. As mentioned previously, the Virtual Notelet system is built upon the GCS. It utilizes low powered, diffuse infrared transmitters as location or physical object identifiers, which can represent entities in physical space. These identifiers are used as indexes to find out the corresponding information about the objects from the context database. The information is sent back to the users through radio wireless LAN.

Client

In order to implement a prototype we had to choose a mobile computing device that is capable of detecting infrared beacons and communicating with the context server to access information using the radio wireless LAN. It could work with a laptop but would be too awkward to use. Similarly, the problems of providing input to and displaying output on many PDAs can also create significant additional barriers to the use of a virtual note system compared to the effort required in order to write a more conventional note and attach it to the user's door. It remains to be seen whether this additional effort will be offset by the benefits that can be provided through the Virtual Notelet. The operation of the application should ideally be as simple and intuitive as using physical annotations instead of forcing users to change the ordinary behaviours. A side effect of the GCS is that it can trace the location of the user's mobile computing devices. Some people reject technology that is able to trace their position or daily routine. Privacy concerns are an important factor that affects the acceptance of such systems. To address these concerns, we ensured that the device's location can only be tracked when the user is actually using the Virtual Notelet application.

Service

It was important that the application not only enable users to author and access virtual notes attached on physical objects but also provide a range of additional functionality that might blend with and support conventional uses of these physical notes. For example, a note might be designed to update their user's location as they move around the environment. Another note might display the individual's published diary from a time management system. In addition, an access control rule/protocol can be applied to address social concerns with the service. For instance, users can only remove virtual notes that were either published by them or attached on physical entities that they "own".

The Virtual Notelet architecture described in this section demonstrates the GCS approach to context-sensitive mobile computing applications. The sensing result or user input can be sent to the remote context server. The client program runs on the mobile computing devices and acts as a collector of context information through existing sensors (i.e. IR port) and displays tailored information according to the current situation. Context information is delivered from the user's mobile computing devices

and interpreted by the server. Further services can be added on the server without impacting existing services. Figure 7.3 illustrates the architecture.

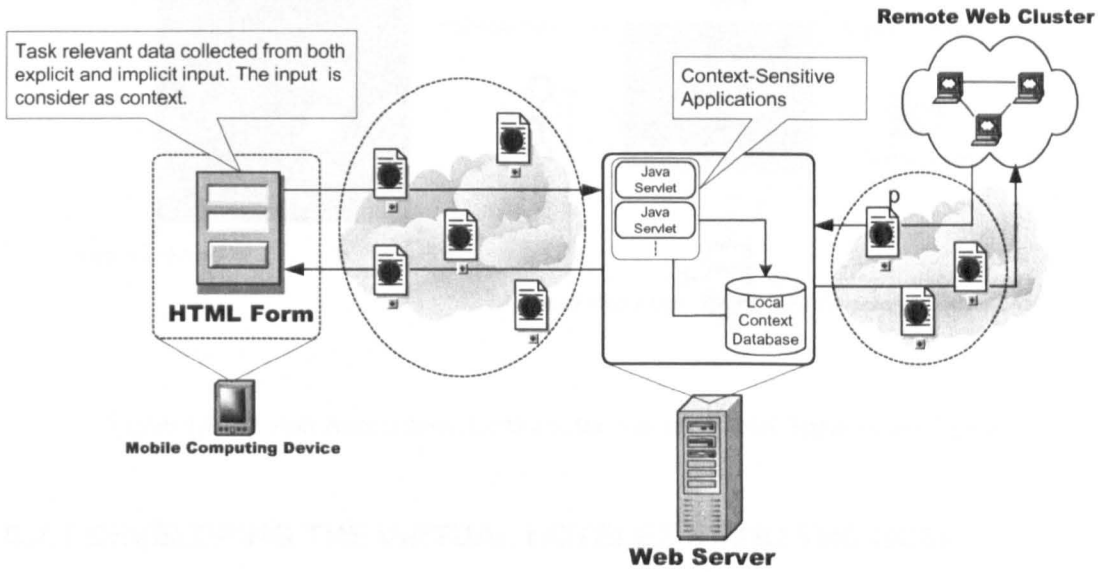


Figure 6.3 The Extensible GCS software Architecture.

6.2 IMPLEMENTATION OF THE VIRTUAL NOTELET

The development of the Virtual Notelet proceeded by identifying a number of scenarios: How can the user see what has been posted to a particular place or object? How can the user post virtual note to a particular place or object? Previous observation shows how people use physical annotations to indicate the status of the occupant in the office and leave messages to the occupant if the meeting is unsuccessful. Physical annotations can be attached to objects, moved with the objects, and seen by anyone near by.

The prototype of Virtual Notelet is implemented to be the system can store virtual notes authored by the users using their wireless-enabled mobile computing devices and display virtual notes attached on physical objects and places on the mobile computing device's screen once they close by. In addition, the system can store the information about who has visited which place at what time. The context information we derived from the task analysis process is the user's identity, location, role, and office occupant's status. Following the design principles mentioned previously, the server is implemented as a Java servlet hosted on a web server and the client is based on Waba. The communication between client and server uses the HTTP protocol. A database is created and implemented in Microsoft Access for storing context information. The JDBC API is used to implement communication between the servlet and the database. Figure 6.4 shows the implementation. The implementation of the Virtual Notelet with respect to the server, client, and service development is described in following sections.

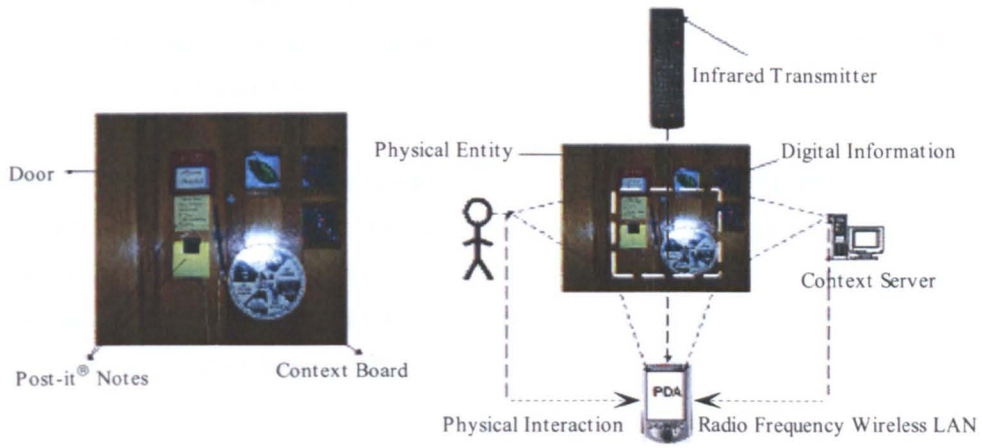


Figure 6.4 Physical Annotations (Left) and the Virtual Notelet Application (Right).

6.2.1 DEVELOPING THE VIRTUAL NOTELET USING THE GCSF

The GCSF follows the Context Toolkit approach, which exploits a "context widget" to gather context information. A context widget acts like a GUI component with access to context information through the use of software or hardware sensors in the system's operating environment. Similar to GUI components, context widgets are associated with correspondent callbacks that are implemented to act upon interaction sequences and trigger the relevant functions once the context changes. In the current GCSF implementation, the RawIR sensor control, which can be considered as a context widget, obtains raw contextual information, infrared identifiers, from infrared transmitters and passes them to an interpreter to convert these infrared identifiers to useful context information for the applications. The idea is:

```
Mouse → device handler → window manager → select events → GUI widget
```

```
RawIR → device handler → context manager → location or physical object
events → context widget
```

We can change the sensor devices but need not change the associated widgets that use the location events:

```
Bluetooth → device handler → context manager → location or resources events
→ context widget
```

```
Differential Radio frequency → device handler → context manager → location
events → context widget
```

The GCSF exploits the HTTP protocol for communication. The eXtensible Markup Language (XML) is also used to wrap sensed context information and specify the relevant function with which to react. The detail is as seen in Figure 6.3.

As noted in Chapter 5, this implementation approach draws on web application development. The benefits of a web-based application are argued to be rapidly evolved with frequent updates and redesign (Gellersen and Gaedke, 1999). The minimal requirements on client side are a technically match for developing applications for the mobile computing devices. Figure 6.5 shows the Virtual Notelet software architecture based on the GCS.

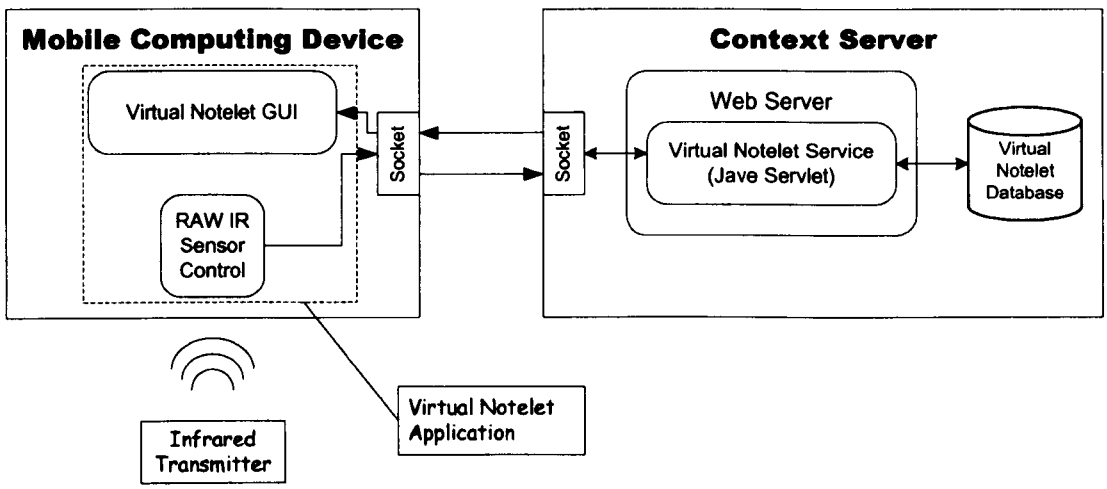


Figure 6.5 Virtual Notelet Software Architecture

6.2.1.1 SERVER IMPLEMENTION

The server implementation is based on Java servlet because this provides a high degree of platform independence, efficiency, modularity, and reusability (Bloch and Bodoff, 2001). A Java servlet is simply a server-side program, which services request and returns responses using the HTTP protocol. This avoids a number of difficulties for communication between the client and the server. For instance, exploiting HTTP can prevent accessing an additional communication port that might be prohibited by a system firewall. Following the HTTP protocol can help avoid the need to implement a proprietary format for reading and writing communication data (i.e. byte stream). Java servlets are similar to CGI (Common Gateway Interface). Without having to create a separate process for each request from a client, the Java servlet simply produces another thread within the same process to deal with each request. The benefit is that the Java servlet can serve more users simultaneously than CGI programmes with less overhead on the server side. The current Virtual Notelet prototype has five different services: user verification (sign up and login), sensor interpreter (translate infrared ID into position ID), virtual note edit (create, attach, and delete) with access control. The relations used in the

server may be reused in a number of different context-sensitive applications. Greater experience in developing these systems should provide greater insights into the potential reuse of these components.

The development of the Virtual Notelet system again illustrated the need to improve the software engineering of context-sensitive systems (Dey, 2000). Microsoft Access® is used to create the context database. This was purely a pragmatic decision and could easily be revised in the light of greater experience. Five tables are built to store the context used in the Virtual Notelet. They are VirtualNotes, Users, Entities, Positions, and UserPosition. The relationship between each table is shown in figure 6.6. Each virtual note has a note ID, message content, author, target entity (where the note attached to), and timestamp. The Users table contains details of each user of the Virtual Notelet. It includes user ID, user name, system login password, office number, and their in/out status. The Entities table stores the digital mapping of physical entities, such as doors, walls, places, objects, and etc. to an ID, name, and position. Each infrared transmitter is treated as an index for mapping the physical entities to their digital format. Information about the position of each infrared transmitter is predefined in the sensor interpreter module, which is one of the base modules in GCS. The position of these transmitters does not support absolute physical positions but the cell-based coverage of infrared transmitters. The infrared signal indexes physical entities in the format "office door – infrared signal" in the Virtual Notelet. A Positions table stores information about the environment into system's coverage. The target environment is room-based partition in this project. The table stores position with the content of position ID and position name. The last table in the context database is UserPosition. As mentioned previously, the users can update their current position by posting a virtual note. It is done automatically by the system. UserPosition consists of user ID, position ID, and timestamp.

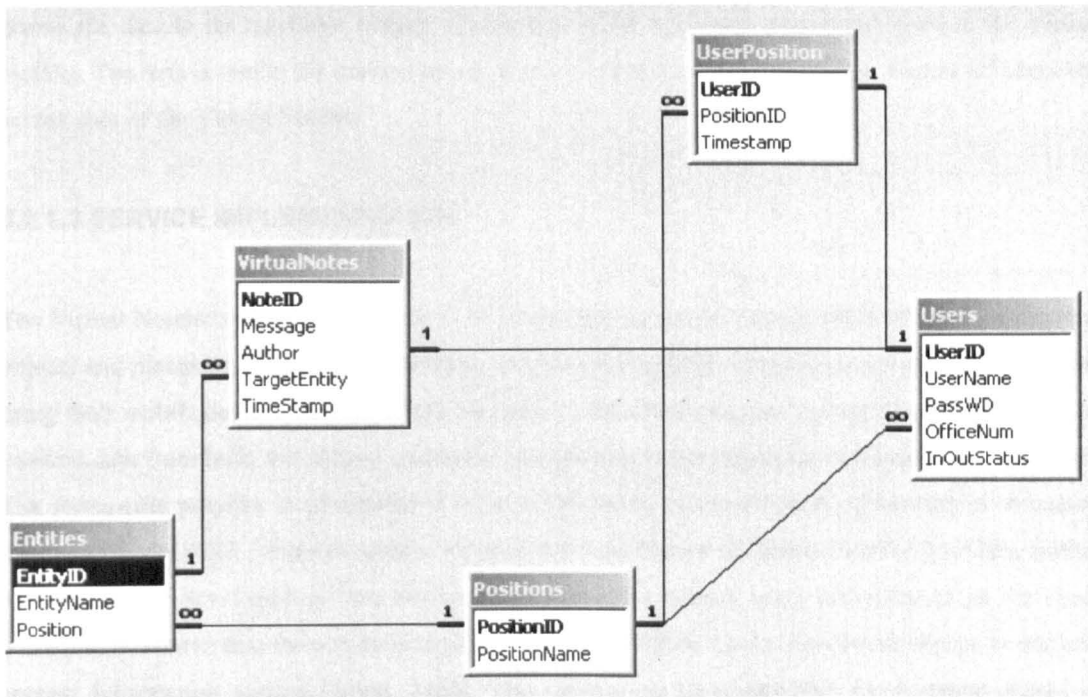


Figure 6.6 Relationships between Tables in the Virtual Notelet Context Database.

6.2.1.2 CLIENT IMPLEMENTATION

The initial system implementation was focused on HP Jornada 680s and HP Ipaqs as target mobile computing devices. The Virtual Notelet application will run on mobile computing devices for which there is a Waba virtual machine. PalmOS devices are currently not supported because of the tight binding with the IrDA protocol as we described in Chapter 5. The context facilities offered by the system will, however, depend on the provision of an infrared transceiver and IEEE 802.11 services. Our target devices satisfied the requirements mentioned in previous section. They are mobile devices that are capable of detecting infrared beacons and communicating with a context server to access information using a radio wireless LAN. These devices provide a touch sensitive screen to display a GUI for users to author and access virtual notes using a stylus. The client program is implemented in Waba. It provides a set of GUI components and a number of I/O classes that includes serial and socket communication support. The client program includes two base modules, which are inherited from the GCS. They are sensor widget and translator widget. The concept of these components is similar to Context Toolkit (Dey, 2000, Dey, 2001). Sensors can be regarded as a GUI component while developing such applications. Once the sensors are triggered, the events are generated from sensor widget and passed to event handler. The event handler may simply receive the raw data from these sensors and pass them to a translator widget in order to convert them into an appropriate format. The translated data are sent to the context server through the wireless LAN. The interpreter API on the server analyzes the data, translates them into context information, and dispatches it to an adequate server-side program (Java servlet). In the case of the Virtual Notelet, the device carried by the users detects the infrared beacons from the transmitters. The sensor (Raw IR) widget posts an event to the handler. The handler activates actions to read the raw data from the sensor (IR port on the device) and passes the data to the translator widget. The format of the translated data is hex code in the Virtual Notelet. The data is sent to the context server as a key to get relevant information. Figure 6.7 show the screen shot of the Virtual Notelet.

6.2.1.3 SERVICE IMPLEMENTATION

The Virtual Notelet system provides the following services: display virtual notes attached on physical objects and places on users' mobile devices, user can edit (author or remove) and attach virtual note using their mobile device. These services are done by the client program collecting explicit as well as implicit data from both the sensors and users and sending to the appropriate context server program. The server-side program implemented in Java servlet has to construct query statements in structured query language (SQL) format to retrieve information from context database. Inspired by Allen, further services can be developed as Java servlets, which either construct query string based on the client context and retrieve data from context database or act as ORB to invoke distributed objects to perform context information service (Allen, 2000). The servlet can be responsible for mapping names to objects, dealing with the lifecycle of object instances for remote clients, and marshalling parameters and return values (Allen, 2000). This forms a much more extensible architecture than the original

GCS environment. In this case, XML may be essential in the client and server communication. Client can use XML to specify the object to call and the regarding parameters. The results is wrapped in XML format and sent back to the client.

The evaluation of context-sensitive mobile computing systems is a research area in its own right. The following sections will, however, describe the field trials that were conducted to support an initial validation of the Virtual Notelet system.

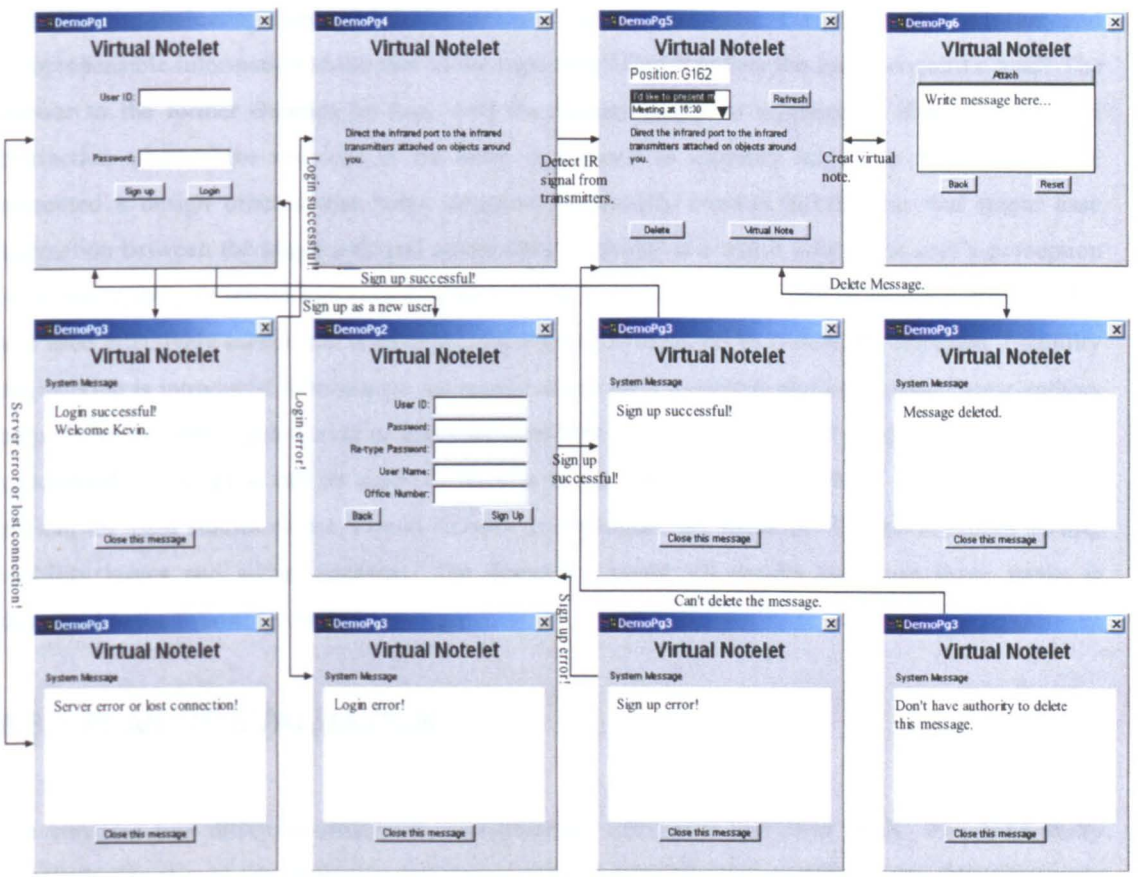


Figure 6.7 The Virtual Notelet Application Screenshots.

6.3 EVALUATION OF THE VIRTUAL NOTELET

In previous sections, we indicated the difficulties in developing a context-sensitive mobile computing system. There is controversy over the definition and use of context information. Also, the development of this type of system requires considerable engineering skills in terms of dealing with the sensors, which are exploited to measure the activities of relevant entities to the interaction and produce results as implicit inputs to the system. The development teams that have the necessary expertise to implement this type of system often focus on innovative system architectures instead of detailed social and usability studies. Some researchers have argued that for this technology to be

successfully applied, it is important that we consider how the system will affect social behaviours (Dryer et al., 1999). The aim of the GCS is to enable HCI designers to rapidly build context-sensitive mobile computing systems using low cost, off-the-shelf hardware and software components. The hope is to help the designers validate the claimed benefits of context-sensitive mobile computing systems without the burden of mastering low level hardware and software techniques.

Scholtz argued that the evaluation for context-sensitive computing systems must judge how well they can reason about context information on a user-centred basis (Scholtz, 2001). In particular, two questions in evaluating context-sensitive systems were highlighted: Can it provide relevant and comprehensible information to the user at the right time? Can it reduce the user's cognitive load? The answer to the former depends on how well the system adapts to information about the ongoing interaction whereas the response to the latter is directed to usability issues. In Appendix B, we presented a design process that helps designers to identify context information that might ease interaction between the target user and application. Usability is a major concern in user's perception of system quality in interactive systems (Butler, 1996). A user interface can be easily comprehended and used effectively carried out if usability engineering is involved in system development. Usability engineering is introduced to minimize the overhead of user's cognition and perception. Some authors argued that metaphor makes easy to teach the user new concepts in terms of ones, which are already understood, if metaphor can be used (Coschurba et al., 2001, Dix et al., 1998). In this section, we present the field studies of the Virtual Notelet. In particular, we focus on the user's pattern of use, usability issues and social concerns. The discussion about the results based on these issues is described in the following sections.

6.3.1 PLAN OF EVALUATION

We consider two different evaluation environments: laboratory and field study. The field study approach was chosen for evaluating the Virtual Notelet because of its mobile nature. The laboratory environment may confine evaluators to predefined situations. It may fail to replicate some cases, such as interpersonal communication tasks. The open nature of the "in the field" evaluation environment can help to address veiled usability and social issues (Burrell and Gay, 2001, Espinoza et al., 2001, Pascoe, 1997).

As mentioned, the initial prototype of the Virtual Notelet system was designed to provide staff and students in the Computing Science department in Glasgow University with the ability to view information about the status of occupants in their offices and to attach/detach virtual notes on places or physical objects in the same way they might use paper-based Post-It notes. We had five participants. Four participants utilized their wireless and Internet-enabled mobile computing devices with the Virtual Notelet application installed. One of the participants did not use a mobile computing device and was provided a PC version of the Virtual Notelet instead. Three infrared transmitters were installed on the doors of participants' office within the department. The duration of the evaluation was

two weeks. This period was chosen because it provided enough time to gather initial feedback on the use of the system. Ideally, a more longitudinal study might have provided additional information. However, as we shall see, such a prolonged evaluation was considered unlikely to yield additional insights.

In order to track the use of the application, a system log function was implemented and used to record the user's activities, such as a user login to the system, writing a virtual note, and posting or removing virtual notes. A timestamp was applied to mark every user activity. As we noted in section 7.2.1.1, the content of each virtual note, which includes the note identifier, author, message body, associated object, and time of creation, is stored in a database located in the context server. The system log and content of each note is the source for us to study the user's pattern of use of the Virtual Notelet application.

6.3.2 METAPHOR SUPPORT

The graphical user interface design of the Virtual Notelet is based on the metaphor of paper-based Post-It notes. Using physical Post-It notes, users can write down a message on the notes, detach and attach them on physical objects. The content of these annotations normally contains purpose, name (author and recipient), and time. There are, however, some important differences between physical and virtual notes. Physical notes cannot determine when to remove themselves or adapt to privacy control. For instance, a note that shows a private meeting at 14:00 may still exist at 17:00 and be seen by passers-by. In the design of the Virtual Notelet prototype, we decided not to automatically destroy obsolete notes. Although paper-based post-it initially appealing, it falls short in the computing world because there are no privacy control equivalents for the physical notes. Applying privacy control on paper-based Post-It notes seems not possible. The virtual notes can achieve this by applying password control on the notes. It is, however, possible to force the users to send private message through traditional approaches such as email instead of attaching virtual notes. They may feel inconvenient to use additional privacy control mechanism to create private messages on virtual notes.

6.3.3 USABILITY ISSUES

As mentioned previously, the design of the Virtual Notelet was intended to provide its users with functionality that required minimal learning through the use of the existing Post-It metaphor. Ideally, it should be as simple and intuitive as using physical annotations instead of forcing users to change the ordinary behaviour. According to the layered design model developed by Bennett, it provides the interaction between real-world tasks, goals, user's operation of the interface, and the system provided function along with user-centred view (Bennett et al., 1989, Butler, 1996). The model supposes that users undertake real-world tasks by applying the combination of system functions and users' cognitive functions. In the case of the Virtual Notelet, users' conceptual model of the work, which is either

viewing the virtual notes or editing and posting virtual notes on physical entities, must be mapped to the users' understanding of the system function. Then, users must build the proper commands to control the system functions needed. Users must physically manipulate the input device (i.e. using stylus to input on touch-sensible screen) to perform the actions according to the commands assigned by users.

Users must interpret the system presentation language in order to know the system status. According to Bennett, users have repeatedly to loop through the various layers of system operations until their goals are reached (Bennett et al., 1989). This means that users may "get lost" in the system if the user interface is badly presented. This model points out the underlying usability issues that may be ignored by the system designers. The difficulties such as learning, retention, error-proneness, etc. may occur because of the mismatches in mapping among each layer. In most cases, user interfaces are correspondent to the model from designers' view of how the system functions were implemented. If the gap between user's conceptual model of how the system related to the real-world works and designer's system implementation model too wide to match. Users have to spend considerable time to translate between and through different layers of operations to achieve the goal of real-world tasks. In contrast, users feel intuitive to use a system if the user interface can provide well-mapped design through all layers. Users are able to predict how the interface will respond and the users' cognitive overhead is not an obstacle between the users and the tasks.

6.3.3.1 MOBILE COMPUTING DEVICE USE

As mentioned, we used HP Jornada 680, 820, and HP Ipaqs. The former (680 and 820) are hand-held devices with an on board keyboard while the later is a palm-size device. They are equipped with a built-in infrared port. The limitation of infrared sensing using the built-in infrared port is that it does not support omni-direction sensing but confines it within a limited cone angle (~30 degrees). This forces users to pay attention to direct their mobile computing device's infrared port to the infrared transmitters. This may affect a user's ability to perform the task of viewing and attaching virtual notes on physical entities. However, in the case of using paper-based Post-It notes, users have to approach the places or objects to view or attach Post-It notes. Likewise, the Virtual Notelet user needs to walk to the office door, which is equipped with infrared transmitters and direct the infrared port on the device to trigger context detection. However, the infrared transmitter installation may affect infrared detection. The position of the infrared port on the mobile device varies with different model. For example, HP Jornada 680 has its infrared port on the left, HP Jornada 820 has its infrared port at the front, and most of palm-sized devices have their infrared port on the top. One of two solutions can be used. Firstly, we might require that the user deliberately angles their PDA so that their infrared sensor is in line with the cone emitted by the transmitter. Some form of auditory feedback might then be used to confirm that a signal has been detected. This violates the notion of implicit location sensing mentioned earlier. Secondly, the transmitter might be located in a position where the signals might be detected from transceivers in several different positions on different PDAs. In order to examine the

feasibility of this approach we modified an infrared transmitter so that the LED generating the signal was on the end of a flexible wire that could be pointed away from the body of the transmitter. Figure 6.8 shows the modified infrared transmitter, which extend the infrared LED out from the transmitter and provided a flexible wire support in order to adjust it to a suitable direction for mobile devices to detect the signal.

As noted in Chapter 5, one of the buttons on the infrared transmitters is taped down to generate a continued output signals. We conducted a test to determine the battery life of each of the transmitters used in the evaluation. Using rechargeable Cadmium batteries, we achieved an average life of six hours. The system, therefore, requires frequently maintenance if we are changing the batteries. The initial stages of the evaluation, therefore, indicated the need to move towards a fixed power supply for the transmitters that would be kept in continual use. The power supply for the infrared transmitters may affect the usability of the Virtual Notelet if the user's mobile computing device cannot detect any infrared signal while standing in front of an office door. This would prevent the user from using the application further.

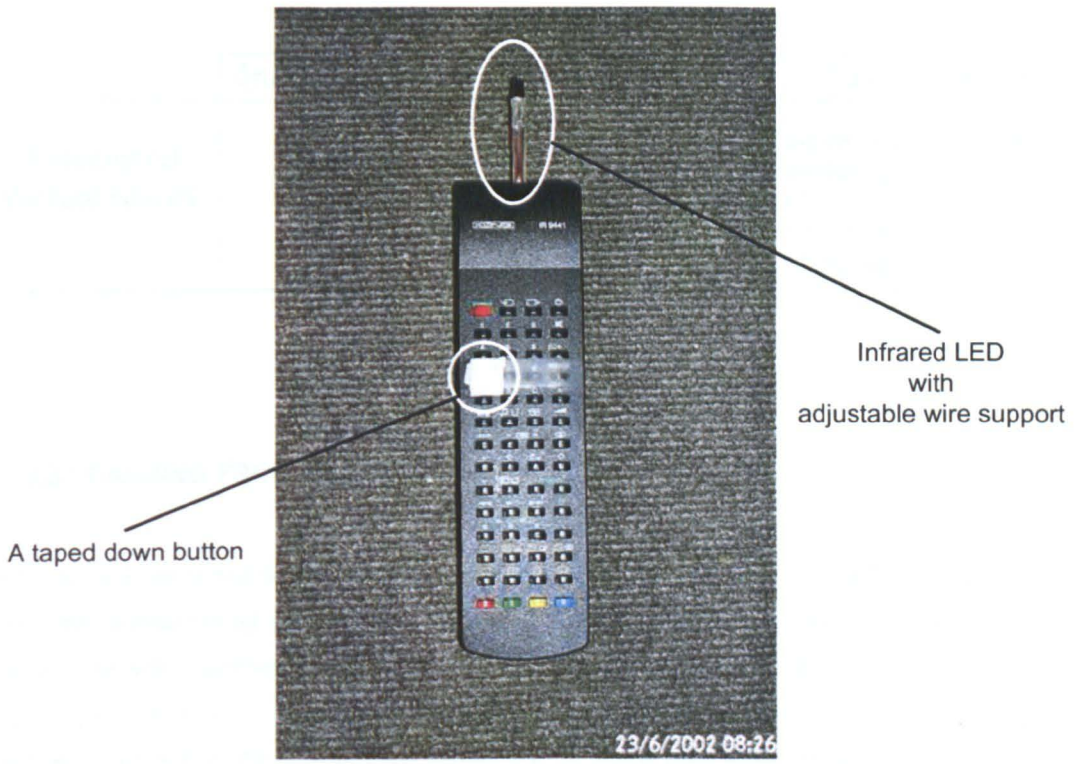


Figure 6.8 Modified Infrared Transmitter

6.3.3.2 APPLICATION USE

The earliest user reaction to the Virtual Notelet was rather positive when we introduced and instructed the users about the operation of this application. The users were enthusiastic to produce test messages

and try the infrared signal detection, which is innovative to them. The log revealed, however, that the users did not produce further virtual notes after their initial attempts. We then decided to initiate a number of virtual notes and attached them on the user's office door. The purpose was to guarantee that the users would see some virtual notes when they used the application. From the result of the system log, we found that most of the users only logged in to the Virtual Notelet twice. The first login happened during system instruction and the second login was to check their virtual notes on their office door and then delete them. Twelve virtual notes were stored in the database at the end of field trial. Five of the virtual notes were posted by us and seven of the notes were created and posted by the users. No notes about the user's in/out status were created. The Virtual Notelet was intended to replace the in/out or context board attached on the user's office door. Most of the virtual notes were test messages. The users created virtual notes only because of the initial attractive nature of the Virtual Notelet system. Other notes shows the Virtual Notelet was used as an email tool. One surprise was that four of the virtual notes were posted by the user with the desktop PC version of the Virtual Notelet. The user sent virtual notes in the way as using conventional email application. However, the user stressed that the privacy concern forced him to turn back to use traditional email application instead of the Virtual Notelet.

	In/Out Status	Context Board	Post-It Notes
Associated Virtual Notes	n/a	n/a	"Hi Steve, I ve placed a copy of the workshop paper under your door."
			"test", "hi kevin", ...

Table 6.1 Virtual note sorts.

6.3.3.3 LEARNING FROM FAILURE

As mentioned, our initial design focused on providing a virtual note service to nomadic users. People could only access virtual annotations if they were standing within range of an associated infrared beacon. This was a deliberate design decision; we were keen to follow the models proposed by the authors cited previously. It was not possible for users in a remote location to determine whether any notes had been left on the door of their office. During initial field trials, we quickly discovered the importance of this facility for many users. Such insights emphasize the importance of conducting field studies to validate the many proposed benefits of context sensitive systems. Our choice of a web-based infrastructure eased the implementation of this facility. Users were provided with the URL for the page associated with their office so that they could retrieve any notes that had been left there wherever they could obtain Internet access. Unfortunately, our response to the users' requests raised a host of security concerns. The location of a user was potentially exposed to anyone with access to the web. We, therefore, implemented fine-grained access control mechanisms based on password

mechanisms. This was faulty avoidable to the desktop PC users. Previous sections have argued that this is a relatively inflexible approach. It can be difficult to temporarily alter access privileges, for instance if a confidential meeting is taking place. Fortunately, all users of the GCS can mask their location at any time by switching off their PDAs. In the future, however, it may be possible to benefit from the more elegant access control mechanisms that have been proposed for successors to HTTP.

Unfortunately, the continuing usability studies revealed a host of further problems that were not addressed by the introduction of these relatively simple privacy mechanisms into the web-based model of interaction. For instance, paper-based Post-It notes provide their readers with a number of different sources of information. They provide the message that is written on them. They also provide an indication that the person who wrote the message was previously in the same location as the note that they have left. This is important because the recipients of these notes often make a number of important inferences based on this information. For example, they may conclude that the writer is nearby if they have left a note in the last few minutes. Unfortunately, by allowing remote, web-based access we not only provided additional facilities to the users of the Virtual Notelet application, we also prevented people from making these forms of inferences. In pathological cases, the system enabled users to deliberately mask their location by leaving messages that indicated they were in one position when they were, in fact, at another.

The field trial of the Virtual Notelet application also identified a number of further usability issues that centred on the naming of particular locations. The initial prototype assumed that the uses would have to be within range of an infrared transceiver in order to read the messages that had been left in that location. This built on the idea that users have to be close enough to a paper-based Post-It note before they can read the information that has been left on it. As we have seen, however, users quickly requested the ability to access their notes from arbitrary Internet connections. They no longer used the infrared beacons as a 'quick index' to the notes that were left in a particular location. We, therefore, developed an interface with a pull-down menu of locations. Users could make a selection from this list to see whether any notes had been left for them in a particular location. Early versions of this system used official room numbers to identify particular locations. Figure 6.9 shows the screenshot of the Virtual Notelet.

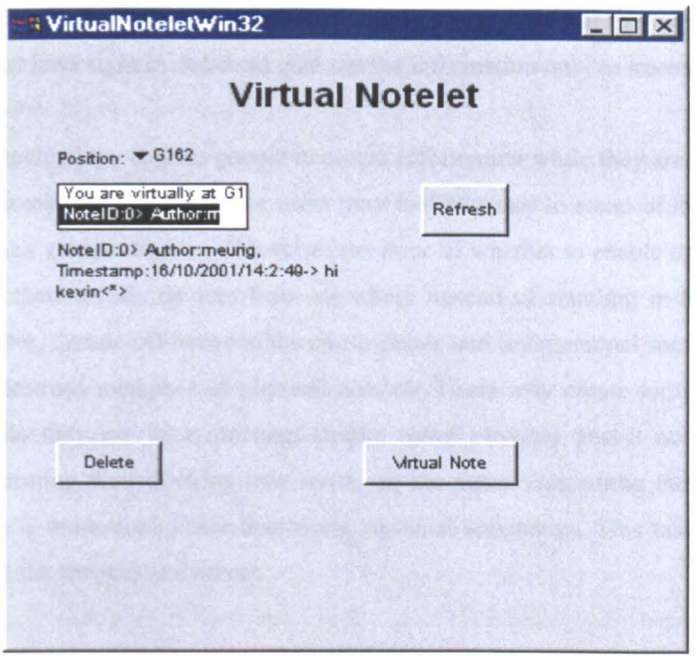


Figure 6.9 The Virtual Notelet with Pull-Down Menu Screen Shot.

This quickly proved to be unworkable because few of the people using the system even knew the official number for their own offices. We changed this to reflect the names that 'most people' used to distinguish the rooms; 'Steve's office', 'the Conference room' and so on. This approach failed to support people who were unfamiliar with the layout of the building. They preferred the official numbering system, which could be decoded to provide an indication of the floor and position of each room. The solution was to enable users to customise the names of the locations that were associated with particular infrared beacons while the system enforced a global naming system that was otherwise hidden from most users. In retrospect, it is surprising that we should not have considered this potential usability problem. In our defence, all that we can say is that such details are easily overlooked during the initial engineering of location sensitive systems. They are also typical of the usability problems that can have a profound impact upon the success or failure of such technology.

6.3.4 SOCIAL ISSUES

The success of emerging information technology will depend on how the technology affects human social behaviours (Dryer et al., 1999). A number of social concerns were mentioned in previous sections. For instance, office occupant can post his current context information on the door for others who walk by the office to decide whether it is all right to enter or leave notes on the door without going in. An access control rule, which determines which user has the authority to remove certain virtual notes, is applied on virtual notes. A side effect of the GCS environment is that it not only can individual devices detect their position using the passive signals that are transmitted by the infrared technology, they can also use the radio network to communicate their position back to the context

server. The system could make available information, for instance, about the user's current position. However, users must have right to decide to give out the information only to trusted recipients.

Mobile computing technology enables people to access information while they are on the move. When using the Virtual Notelet application, some users may feel reluctant to stand in front of an inanimate object consulting their mobile devices. This raises the issue of whether to enable users to read and post virtual notes using their mobile devices from anywhere instead of standing in front of the specific objects. It is, however, a trade-off between the convenience and interpersonal social effect. More of a problem is that it destroys metaphor of physical notelets. Users may create virtual notes using their mobile device while they are in a meeting. Unlike using physical post-it notes, users only pay attention on transforming their thinking into words on the notes. Consulting their mobile device to create virtual notes is more complicate than using physical annotation. This task may distract their attention away from the speaker and others.

When a new technology appears, it is possible misused that leads to unexpected results (Dryer et al., 1999). For instance, people can defame others by writing something unfavourable using physical post-it notes and attach them everywhere. It is likely to happen in using Virtual Notelet application. Someone can create virtual notes that contain slanderous content and post them anywhere s/he wants within the system-hosted environment. It is, therefore, we apply a user login mechanism to guarantee each virtual note has its correspondent author. This protection, however, cannot be promised unless there is a security control mechanism that cannot be broken by hackers existing.

6.4 SUMMARY OF THE VIRTUAL NOTELET CASE STUDY

This chapter has described the design and implementation of the Virtual Notelet project. The GCS has been exploited as the base architecture for this case study. The Virtual Notelet system follows the second implementation approach of the GCSF. It utilizes Web technology to enable users to access context-sensitive services based on a heterogeneous platform. The GCS has the advantages of modularity and extensibility. Further context services can be added using Java servlet without impacting existing services. A context-sensitive mobile computing application cannot hope to match the flexibility of the physical medium. The ability to hold, fold, interleave and manipulate paper as a physical medium is very difficult to recreate in the direct manipulation dialogue of current interfaces. However, future versions of the Virtual Notelet can be extended to offer a range of additional functionality that might blend with and support the conventional uses of these physical annotations. For instance, a note that the users could update with their locations as they move around a building can be added. Another note might display the individual's diary from a time management system that was already being used within our target organization.

We used this Virtual Notelet prototype to conduct an initial field study. This raised many questions about the methodology that might be used to evaluate this new generation of context-sensitive mobile

computing systems. For example, is it possible to overcome the initial enthusiasm that many users express for novel applications on PDAs, especially when there may be significant usability problems affecting the long term success of an implementation?

Many of the design concerns that arose in the design of Virtual Notelet had little to do with the underlying communications technology, which is often a focus for work in this area. In contrast, we were interested in social and personal usability concerns. For instance, a current problem with existing notes is that they are visible to anyone who might pass a particular door. Users were concerned to associate password protection or other permission mechanisms to the notes that they left. Other concerns related to the longevity of a note.

"More importantly, the initial design activities were dogged by a form of skepticism that is not recognized by many of the more euphoric attitudes expressed by the proponents of context-sensitive mobile computing systems. The earlier proponents of this technology had stressed the benefits of virtual Post-It notes. However, they often failed to recognize the pragmatic barriers. Many people did not posses PDAs. Others forgot to carry them with them or simply did not attempt to access the digital information. Very importantly, an infrared transmitter used in Virtual Notelet lacks the affordances provided by a physical note, whereas touch sensitive screens in public areas can address some of these problems (see figure 6.10). However, this again raised security concerns and is far from an ideal solution. In addition, the affordance of having digital information associated with locations and physical objects may not be clear and should be briefly explained. One should also be aware that many users have no experience of using a mobile device before and are unfamiliar with the differences between a conventional desktop PC and a mobile device. It is important to avoid unnecessary features that may only confuse users" (Cheng and Johnson, 2001).

During the field trails, we also found out that our users were familiar with a host of more conventional communications media such as physical notes and electronic mail. They had developed expertise in using these applications to support their daily tasks. It is, therefore, difficult to persuade users to continue using the Virtual Notelet application. However, this does not mean that context-sensitive mobile computing systems offer few benefits to their users. Our findings do, however, confirm the importance of obtaining direct evidence from field trials before making more elaborate claims about the potential benefits of context-sensitive mobile computing. A preliminary conclusion from this work is that context-sensitive mobile computing systems not only depend on a revolution in their underlying technology but also require fundamental changes in the way in which people access information as they move around their physical environment.

Figure 6.10 Touch Sensitive Screen V.S. Infrared Transmitter.

Some of the user interface of the Virtual Notelet did not satisfy generally accepted HCI design principles. For example, the text field for data entry should be aligned in a jagged fashion to reflect the fact that each field name and its content is of different lengths (Dix et al., 1998). However, the emphasis on this prototype application is to demonstrate that the GCS is functioning and can act as a test-bed to host applications, such as Virtual Notelet in this case. Further development and improvement could be carried out to address the current user interface issues.

In the next chapter, we will discuss the design of a second system that was used to test the GCS environment. This focussed on the provision of information about products and services within a large "superstore". This move from the more conventional office based, "follow me" applications was deliberately intended to pose new challenges for the architecture. The intention is also to see whether other programmers can use the GCSF API to build a context-sensitive mobile computing application.

CHAPTER 7
CASE STUDY 2: MAPVIEW SHOPPING ASSISTANT

In order to demonstrate the extensibility and flexibility of the GCS environment, we issued the development package, which includes the off-the-shelf hardware and the software development API, the GCSF, to two final year students in our department to build a shopping assistant application in a shopping mall domain. They secured the support of a hardware "mega-store". The intention was to determine whether other designers with a basic grounding in computing might be able to use the services provided by the GCS environment. The application, MapView shopping assistant, has been built using the GCS environment. As we mentioned in Chapter 4, the GCS can be utilized to implement location-sensing, location-disclosing, and environment-sensing systems. The goal of this application is to evaluate whether the GCS can be implemented in warehouse-style retail stores and exploited to address particular issues, such as customers getting lost and requiring assistance, in this type of environment.

Most paper-based store guides already offer their customers, for example store layout features that can guide their shopping. However, it cannot adapt to changes as a context-sensitive mobile computing application can. Our initial idea was to provide the customers with an application, which displays a personalized map, in order to orient them within the building and locate products and other services on a map. The application is expected to provide fine-grained information by zooming in to the place of interest. It can notify the user about the products around the current location. It can also provide inventory information to staff. In addition, the customers' locations can be obtained by the application while they are shopping around within the store. The studies of how the store is used by customers and how to optimize the layout for maximum profitability on a store-by-store basis can be derived from the information about the customers' locations (Asthana et al., 1994, Chan, 2001a). This is beyond the scope of the evaluation of the GCS environment. However, the example simply shows rich potential in discovering research issues in a context-sensitive mobile computing environment. Our focus is on the MapView shopping assistant design process mentioned in Appendix B and how the application was built upon the GCS environment.

7.1 APPLICATION DESIGN

In the design of the MapView context-sensitive mobile computing application, the priority of the application is to keep the customer informed about her current location, what products are nearby, and how to get to the product that she is willing to buy from the current location within the store.

7.1.1 APPLICATION DOMAIN AND SCENARIO

The aim of this application is to provide customers with the following shopping experience: a customer wants to buy a particular product in the store. She initially examines the contents of the current area and finds out that the product is not placed locally. She can check the product list and examines the details available on the product using the application. The application can inform her about the stock availability of the product. Right now, she is informed of the product's presence and wishes to locate it. The application can display the information about the customer's and the product's location on an electronic map. This map is updated as she moves towards, or away from, the product.

The students focused on the example scenarios described above. Similarly, they exploit the task analysis technique adopted in the previous chapter to analyse the different situations in terms of the user's use of a paper-based store guide and the context-sensitive shopping assistance application.

7.1.2 USER MODEL

The core function of the MapView is to help the customer navigate within a store and finding products. As we noted in Chapter 2, some previous approaches in this domain utilize the customer's shopping history about what she bought when she came to the store last time to provide a personalised shopping experience. There is rich potential for user models in this domain (Asthana et al., 1994). However, our intention is to demonstrate whether the GCS environment can be used by other designers to implement applications in this domain. The students simply focused on two different types of users; customers and staff members. More detail about the customer's tasks shopping in the store will be described in the next section. However, the application scenario about the staff member support has not been implemented and remains for future development. The reasons for this are mainly due to resource constraints and will be discussed in the following sections.

Unlike the Virtual Notelet application domain, warehouse-style retail stores are visited by a wide ranging cross section of the general public, with a wide range of computing ability spanning the complete spectrum from absolute novice to expert. The Virtual Notelet was designed for use by people who already have a PDA so they may be better able to understand the interface to a context-sensitive mobile computing system. The initial design scenario for the shopping system was that people who walk into a shop could be given one of these systems. As we shall see in the evaluation of the field trial carried out in the real store by the students, customers are very unlikely to be familiar with mobile computing technology. The lack of experience with mobile computing technology makes it even more important that any context-sensitive mobile computing service does not impose additional overheads on them. Therefore, it is important that the map interface is simple, since it is to be used as a primary source of navigational information. In addition, since providing product information is another function, the transition between using the system for navigation and using it for

browsing product information must be transparent. Most users will be using the system to retrieve location and product information quickly and will not want to spend time learning how to use it.

7.1.3 TASK ANALYSIS

We express the potential situations that would happen in the scenario and exploit the HTA and entity-relation-based task analysis techniques to identify context information for the application in each situation.

Situation 1: a customer looks up a store guide and wants to know her current location within a shop using a store guide

High level view of context: a customer looks up the store guide.

Role: customer

```
0. in order to identify the current location in the shop
      1. enter a shop
      2. obtain a store guide from the helpdesk
      3. glance round the surroundings
      4. look up the store guide

Plan 0: do 1-2-3-4 in that order
        iteration between 3 and 4 may be necessary to identify the current
        location
```

Object customer **human actor**

Actions:

```
C1-1: enter a shop
C1-2: obtain a store guide
C1-3: glance round the surrounding
C1-4: look up the store guide
```

Object store guide **Simple**

Attributes:

Affordances: hold/fold/mark or draw

Relations: object-object

```
Location (store guide, helpdesk or store entrance)
```

Relations: action-object

```
patient (C1-2, store guide)
```

- a customer obtains a store guide

patient (C1-4, store guide)

- a customer looks up a store guide

Relations: action-event

before (C1-2, C1-4)

- must obtain a store guide before use it

A default view of the map showing the entrance and exits is displayed to allow users to orient themselves as they first enter the store. The main features of the store are pointed out to give a brief overview of the type, size and services of the store. Now we look at how the MapView application helps the user in this situation.

Object MapView **non-human actor**

Actions:

MV1-1: display the store layout

MV1-2: display the current location

Object virtual map **Simple**

Attributes:

Affordances: browse (up/down/left/right)

Relations: object-object

Location (virtual store guide, mobile computing device)

Input to MapView

customer holds up and takes a look at the PDA → MV1-1

customer holds up and takes a look at the PDA → MV1-2

Context in MapView

[user location] and [user movement] → MV1-1 and MV1-2

The MapView recognizes contextual communication through the customer's movement. When the customer holds up his PDA and looks at the screen we assume that he would like to know his current location and information about the store layout.

The following describes the situation that a customer wants to go to specific products that interest him.

Situation 2: a customer looks up the inventory section and wants to know where to get to the product that interests him using a store guide

High level view of context: a customer looks up the inventory section on the store guide

Role: customer

```
0. in order to know the direction to the interested product
        1. look up the store guide
                1.1 check the location of the interested product
                1.2 identify the current location
        2. glance round the surroundings
        3. walk to the product

Plan 0: do 1-2-3 in that order

Plan 1: do 1.1-1.2 in that order
```

Object customer **human actor**

Actions:

C2-1: look up the store guide

C2-2: glance round the surroundings

C2-3: walk to the product

Object store guide **Simple**

Attributes:

Affordances: hold/fold/mark or draw

Relations: object-object

Location (store guide, helpdesk or store entrance)

Location (store guide, customer)

Relations: action-object

patient (C2-1, store guide)

- a customer looks up a store guide

Relations: action-event

before (C2-1 and C2-2, C2-3)

- must obtain the direction to the product from current location using a store guide before walk to the product

Object MapView **non-human actor**

Actions:

MV2-1: display the store layout

MV2-2: display the current location

MV2-3: present an inventory look up function

MV2-4: display the location of the product

Object virtual map **Simple**

Attributes:

Affordances: browse (up/down/left/right)

Relations: object-object

Location (virtual store guide, mobile computing device)

Input to MapView

customer holds up and takes a look at the PDA → MV2-1

customer holds up and takes a look at the PDA → MV2-2

customer select inventory look up function → MV2-3

customer choose the product → MV2-4

Context in MapView

[user location] and [user movement] → MV2-1 and MV2-2

[interested product] → MV2-3 and MV2-4

Users may wish to view information about a specific area in the store without physically being located in that area. This may be helpful when the product is not available in the current location. The user is specifically interested in one product and relative to their current position, requires directions to find the product. The following HTA describes the solution without the mobile computing system before we describe the HTA for the implemented system.

Situation 3: a customer fails to use the shop guide in situation 1 and 2 and seeks for the help from the staff nearby.

High level view of context: a customer cannot identify his current location and do not know how to get to the product that interests him

Role: customer

0. in order to find help from a staff
    1. glance round the surroundings
    2. find a staff nearby
    3. walk to the staff
    4. explain the request
    5. follow the staff's suggestion

Plan 0: do 1-2-3-4-5 in that order

Object customer **human actor**

Actions:

C3-1: glance round the surroundings

C3-2: find a staff nearby

C3-3: walk to the staff

C3-4: explain the purpose

C3-5: take the staff's suggestion (identify the current location or direction to the interested product)

Object store guide **Simple**

Attributes:

Affordances: hold/fold/mark or draw

Object member staff **human actor**

Actions:

MS1: actively look for customers who are in need of help

MS2: converse with the customer and understand the requirement

MS3: perform help

Relations: action-object

patient (C3-4, staff)

- a customer asks a member staff for help

Relations: action-event

before (C3-2, C3-3, C3-4, C3-5)

- must find a staff before asking questions

Object MapView **non-human actor**

Actions:

MV3-1: display the store layout

MV3-2: display the current location

MV3-3: present call assistant function

MV3-4: display the location of nearby staff

Object virtual map **Simple**

Attributes:

Affordances: browse (up/down/left/right)

Relations: object-object

Location (virtual store guide, mobile computing device)

Input to MapView

customer holds up and takes a look at the PDA → MV3-1 and MV3-2

customer select "assistance" function → MV3-3

customer holds up and takes a look at the PDA → MV3-4

Context in MapView

[user location] and [user movement] → MV3-1 and MV3-2

[user fails to orient himself] → MV3-3

[user location] → MV3-4

To summarise, the context information that we elicited from the task analysis for the MapView application is: user's location, user's movement, location of interested product, and failure in orientation. In the example scenario, we assume that the user will initially be motivated to examine the products listed near their present location. As we shall see, however, the field trials indicated a variety of alternative uses for the system. The results of the requirements gathering from the interview carried out by the students with managerial member in the store show that quite a few customers ask for help from the member staff while they shopping in the store. The major questions are most started from "where is...?", ranging from the particular products to facilities in the store such as the toilets. This echoes our emphasis in location context in this application. However, they also often approached staff with supplementary questions for advice about the use of a product. Further details about the requirements engineering for implementing the MapView in the store should refer to (Gibson, 2002).

7.1.4 OPERATING ENVIRONMENT

The results of the requirements engineering carried out by the project designers suggested that the infrared sensing used in the GCS environment is suitable for the layout of the store. A number of 9 feet tall bays contain products and form the layout of the store. The distance between product bays is sufficient to avoid infrared signal interference if two infrared transmitters are placed too close to each other (i.e. within 30 degree cone). In addition, each product in the store has been assigned a product code that indicates its category and identity. Each product bay is associated with a unique identifier.

The MapView's operating environment comprises of infrared transmitters placed around the store and a set of infrared enabled PDAs. The infrared signal is exploited to index the store in terms of product bays and items. The signals also infer the location of PDAs when they detect the infrared signal. The store used as a pragmatic case study is equipped with a radio wireless LAN for stock control. Considering the current memory limitations of the PDAs and amount of products in the store, the issue of storage is needed to be taken into account. For the purpose of this prototype, the number of products is sufficiently small to be stored on the PDAs themselves. However, the PDAs would have to follow the full GCS environment, which is equipped with radio wireless LAN support, in the principle system in order to have access to remote information about products. This data is too large to fit into the memory on the PDAs. The PDA can also provide information about stock levels as they change over the day. Whenever the PDA detects a different infrared signal, this would trigger a request over the radio wireless LAN to the database for the relevant information in a similar manner to that currently employed using a barcode system (Asthana et al., 1994). In addition, the wireless LAN could support communication between the customer and a member of staff. In this situation, a message that contains the received infrared signal identifier can be sent to the server over the radio wireless LAN. The server side program then translates the infrared signal into location information. The appropriate assistant is dispatched to the customer who needs help. A similar approach has been implemented to assist tutor-students interaction in teaching (Crease et al., 2001). However, there were too many commercial and security concerns to allow direct access to this radio wireless network

during the field trials. Therefore, the MapView prototype simply follows the indoor Cyberguide approach, which pre-caches all the location dependent information in the user's mobile computing device (Long et al., 1996).

7.1.5 ACTIONS

The prototype designers exploited context information to implement and perform the actions mentioned in the previous section. These actions are "display the user's location and relative store layout", "display the location of interested product", and "call nearby assistant". All actions are implemented in MapView and executed on the PDA except for the request for staff assistance due to the lack of wireless LAN access. Some sample test data with the format of information stored in a stock database was exploited in the current implementation. The data is simply stored in a text file with format shown as follows:

```
Location or Product Section: Description: Location–Beacon Identifier
```

This is similar to the Virtual Notelet's "lookup table" approach. The Virtual Notelet's lookup table is stored in the remote context server while the location or product-infrared beacon mapping data in the MapView resides in the user's PDA. This file is simply transferred on the device for testing. Once the customer's PDA detects the infrared signal, the signal is then translated into location or product information displayed on the PDA's screen.

7.2 IMPLEMENTATION OF MAPVIEW

The major context of the MapView application is the location of either the user or products within the store settings. The MapView performs the context-sensitive behaviour that allows the user to keep track of their location relative to their environment at all times, as well as to allow the user to locate and observe details about each product in the store. The decision was to create a navigational store map using static images based upon the HouseView application described in Chapter 5.

MapView is divided into two parts: Presentation of information about products and user's location on the PDAs and user's location and nearby product sensing. Location and environment (i.e. products) sensing results are passed to update the plan showing on the user's PDA and list of the products in the vicinity.

7.2.1 DEVELOPING MAPVIEW USING THE GCSF

As we described in the Chapter 5, the GCSF API provides the designers with a sensor control interface. The designers can implement their sensor control and "paint" it on their application. The current MapView implementation adopts the same RawIR sensor control as the Virtual Notelet

application. The RawIR sensor control can generate a mouse-press like event once the infrared transceiver on the user's mobile computing device detects and identifies the incoming signals from the infrared transmitters deployed around the user's immediate environment. The GCSF does not provide an API that helps the designers to display the location and environment context. The following section describes the challenges that the designers encountered in implementing information presentation for the MapView application.

This focuses on the choice of programming language. GUI support was an important factor in implementing the MapView application. However, GUI support is poor in Waba, which we used to implement the GCSF API. The designers, therefore, decided to adopt SuperWaba to implement the MapView. SuperWaba is a Waba based programming language with more GUI components. This approach should have supported the presentation of context information in the MapView prototype. The following figure 7.1 shows the MapView software architecture based on the GCS.

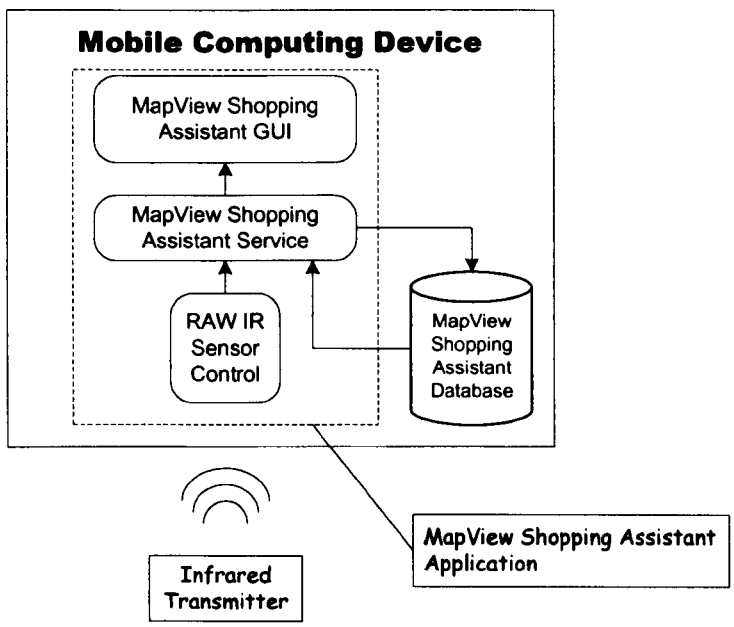


Figure 7.1 MapView Shopping Assistant Software Architecture.

7.2.1.1 INFORMATION PRESENTATION

The reason for adopting a map to visualise the user's location was not only based on the task analysis but also on the results of requirements engineering discussions with the manager of the mega-store about the most common customer queries on locations. As mentioned, the information presentation part of the MapView consists of two segments, mapview and productview. The mapview segment displays a floor plan of the store and the customer's current location. A productview shows information about products in the nearby area.

The first challenge was to provide an efficient means for the user to manoeuvre their view of the floor plan on the PDA. As the user's primary means of interacting with the application was to use the stylus

to manipulate the current view by centring the portrayed location upon the stylus pressed coordinate. After a brief field trial done by the prototype designers, it was discovered that the ability to interact with the map display was not apparent without explicit explanation. To address this predicament, some visual aides in the form of directional icons on the borders of the display were used. In addition, in order for the users to keep tracking their current location, a visual representation of themselves on the electronic map is exploited as the text message "you are here!" in the HouseView application described in Chapter 5. This update process on the electronic map is performed every time the user leaves one location for another. The map automatically centres the user's position and displays a red person icon which is clearly defined from the white floor plan as depicted in figure 7.2a. An auditory notification was used to inform the user of the "context" change. This approach can help to tackle the sensing problem as we noted in the previous chapter. The advantage of being given the user's current location in a context-sensitive mobile computing system in a shopping mall domain is that the user will instantly know where they are as opposed to manually mapping the surroundings onto a paper map.

The next challenge is to help the user to retrieve the location of every product within the store with minimal effort. In the paper-based store map, we assume that the user will initially identify the item name from a product list and then look up the map for the associated store section. This is a slow process while the user is shopping in the store. In the current implementation, the user must first use the system interface to select the product inventory. In order to locate a particular product entry from the large volume available, an initial filtering was designed using techniques found on web sites containing large volumes of data. The user is presented with the initial letter of the product name. Figure 7.3 illustrates this approach. Problem can arise if many products have the same initial letter. Similarly, users may not know the precise name of the item they are looking for. Also products may be filed by purpose, for example, "Saw" or manufacture "Stanley". This filtering technique could be revised in the future versions. Once the product is located, the user can choose the mapview element which has now highlighted the location of the product using a star icon as shown in figure 7.2b.

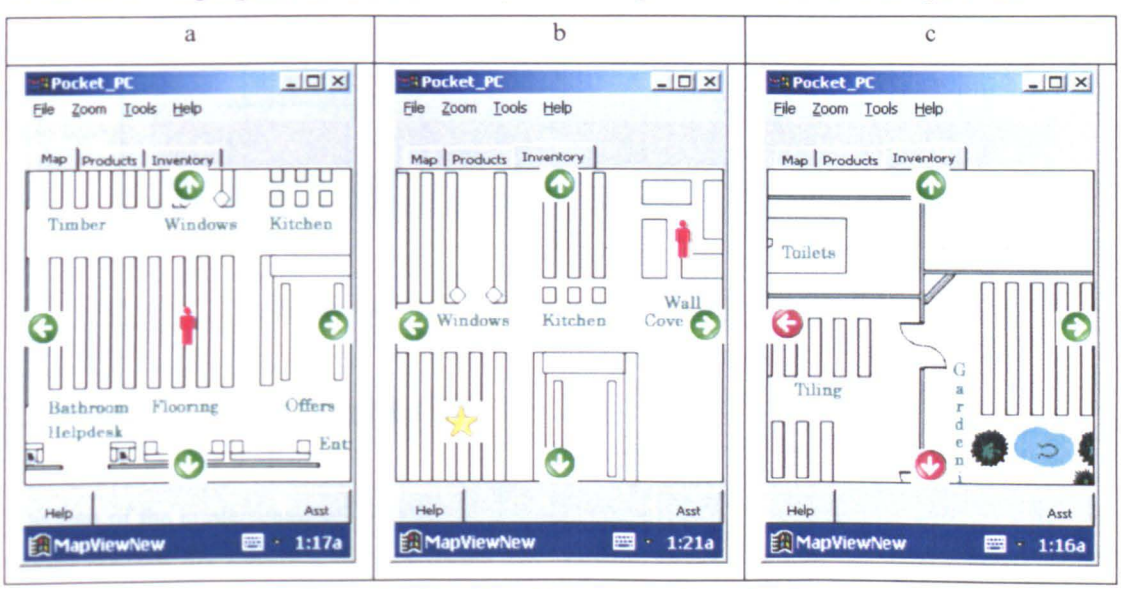


Figure 7.2 MapView Screenshots (Scott, 2002)

As can be seen in the figure 7.2b the user is relatively close to the product. What if the product is located at a great distance from the user? This problem could perhaps be solved with the option of scaling the map out to include both the user and the product. However, the lack of image manipulation APIs in SuperWaba makes this difficult. Instead, the designers used directional arrows along the border to show the user the products' location without having to traverse the map image. As illustrated in the figure 7.2c, the current view has been adjusted with highlighted red arrows on the left and bottom to indicate the direction to scroll the map in order to locate the product. The user can be guided from their current position to the product simply by following the arrows highlighted on the device.

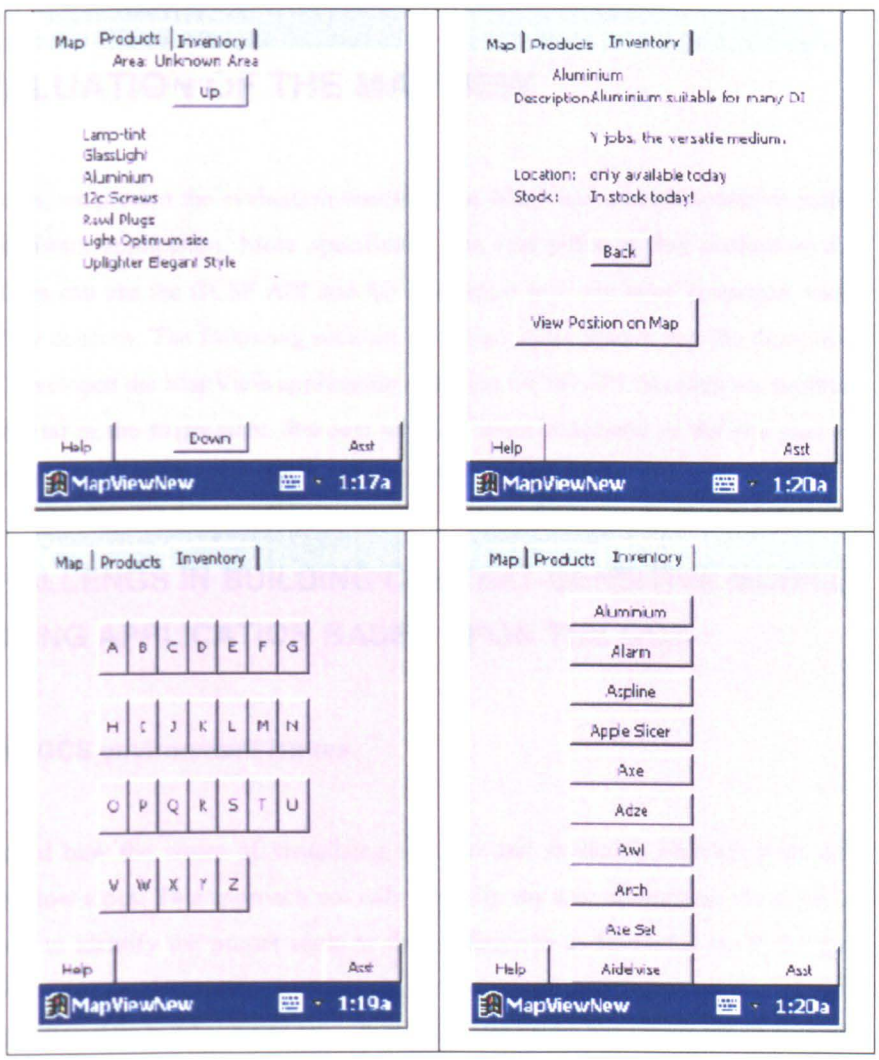


Figure 7.3 Productview Screenshots (Gibson, 2002)

7.2.1.2 ENVIRONMENT INTERACTION

This part of the implementation inherited the existing RawIR sensor approach described in Chapter 5. Sensor controls are regarded as UI modules, such as GUI components, in the GCSF. Based on the current implementation of the GCS, we "paint" the rawIR sensor component in the main interface

instance as this is the panel which is to be displayed. Whenever an event is generated from within this main panel it is sent to a suitable handler for "updating". We also incorporated audio into the interface as mentioned in Chapter 5.

Each location within the store is indexed by an infrared identifier. Each received infrared signal of a corresponding location is found then the map interface will be notified of the update in location. However, a number of problems occurred during the implementation of this portion of the MapView in terms of the implementation language. The details about these issues are described in the following section.

7.3 EVALUATION OF THE MAPVIEW

In this section, we present the evaluation results of the MapView context-sensitive mobile computing shopping assistant application. More specifically, the real point of this evaluation it to determine whether others can use the GCSF API and the evaluation with the store customers was an important but secondary concern. The following sections first discuss the issues that the designers encountered when they developed the MapView application using the GCSF API. Second, we summarise the result of the field trial in the mega-store. Readers who are more interested in the end user experiences in using this type of system should refer to (Gibson, 2002, Scott, 2002).

7.3.1 CHALLENGS IN BUILDING CONTEXT-SENSITIVE MOBILE COMPUTING APPLICATION BASED UPON THE GCS

7.3.1.1 The GCS environment Issues

We mentioned how the issues of visualising the user and product's location were addressed using directional arrow icons. This approach not only can help the user manoeuvre the store layout but also help the user to identify the proper route to the product she is interested in. If the product is some distance from the user's current location these directional arrows along the border of the map highlight the direction to take. However, what if the user becomes disoriented? The electronic store map in the MapView is only oriented correctly when the user is facing the back wall of the store. Because the infrared sensing used in the GCS cannot work out which direction the user is facing. The infrared signal simply indexes locations or products in the store. As we mentioned in Chapter 3, a number of existing location sensing techniques, such as the Active Bat and RADAR can address this issue. The underlying concept of their approach is to measure the changes of the signal factors (i.e. strength or time-of-fight) and these changes could be used to determine the user's orientation. Other solutions could be providing the users with an orientation option so they can choose which orientation they are currently facing. The map is then rotated to suit the user's current orientation (Crease et al.,

2001). Similarly, GPS units can provide a compass to ensure that the top of the map is actually what the user is looking at even when they turn around to look at something else in the same infrared cell.

7.3.1.2 Programming Language Issues

As mentioned, SuperWaba was used rather than standard Waba because it provides greater GUI support. Before carrying out the implementation, we checked the SuperWaba official website for information on supported devices and it stated that PocketPC (formerly known as Windows® CE) devices were fully supported up until the latest versions of the SuperWaba virtual machine. However, this was an error due to website misadministration. This created major problems in language compatibility as the project developed.

As we noted the MapView application is divided into two parts, context information presentation and infrared sensing. The designers were able to construct the entire interface using the GUI elements within SuperWaba. They evaluated its performance using a Java appletviewer execution of the code within a desktop environment. The infrared sensing for the MapView using the GCSF could not be emulated on the desktop computing environment. They had to test the program on the target mobile computing device, a HP Jornada 565. However, during integration, it was discovered that the various GUI elements incorporated into the interface design were not available on the target device. The reason that the issue was not highlighted earlier was because the SuperWaba web site stated that methods developed to run on the SuperWaba VM will happily run on the standard Waba VM. To tackle this issue the MapView designers contacted the SuperWaba chief developer.

"Guilherme Campos Hazan the chief developer of SuperWaba replied with a small apology with regards to the anomaly on the website leading to the erroneous choice of implementation language. Also appended was a statement as to his intent to concentrate his efforts towards the Palm Pilot device, largely due to a greater commercial interest in developing Waba programmes on the this platform, he did however state his intention to find a solution to supporting Windows CE device platform in the future once time allowed." (Scott, 2002)

As mentioned previously, the first step to build the MapView application was to acquire a real map to base the electronic map upon. The second was to supply the electronic store map file to the program and manipulate it using SuperWaba. This, however, was not possible due to the incompatibility problem with SuperWaba. Standard Waba provides programmers with more restricted graphical manipulation routines. This limited the intended visualisation options regarding the map. It was not possible to scale the mapview dynamically from within the programme without reprocessing the map image file to represent the zoom level. Another obstacle was that the SuperWaba virtual machine for Windows CE devices does not support audio feedback. This experience showed us that providing a

GCS platform does not solve all of the problems in building context-sensitive mobile computing applications. The meta-level message here is that even Java style virtual machines may not be as portable as we thought.

7.3.2 SUMMARY OF THE FIELD TRIAL RESULTS

The evaluation of context-sensitive mobile computing applications raises challenging issues. The approach to this particular evaluation was based on measurements of user feedback and behaviour. Questionnaires, focus groups, direct observations and talk aloud sessions were used in order to obtain the results. As we mentioned previously, due to the mobile nature of the context-sensitive mobile computing application, the evaluation must take place within the intended environment. The relationship built throughout the project development between the prototype designers and the manager of the field trial environment in a mega-store created an opportunity to conduct some evaluations within the store amongst "real customers".

7.3.2.1 PLAN OF EVALUATION

A comparison experiment was carried out to evaluate the usability and usefulness of the MapView application. A series of simple tasks, which could be carried out using the paper-based store guide, and the MapView application, were arranged. The participants would then answer the questions using both tools and rate how they perceived each approach. A questionnaire was constructed. Initially it was intended to find out about how often the participants use information technology in everyday life and general attitudes to context-sensitive mobile computing technology. In order for the participants to gain a basic impression of the use and limitations of each method, participants were instructed and asked to perform the tasks using the paper-based store guide to answer some questions, and using the PDA to answer some other questions were carried out subsequently.

7.3.2.2 METAPHOR SUPPORT

The feedback from the direct user observation section identified a regularly occurring problem that caused the participants to appear confused and frustrated. User had difficulties in interpreting the designer's use of the choice of colours for the directional arrows on the MapView graphical user interface. In the current implementation a green arrow means that the current product that the user is looking for is within range of the screen. In contrast, a red arrow means that the location of the product is so far away from the user's current location that the map cannot display them in the same screen. However, problems occurred because people used the common association of traffic light colours with proceeding in a particular direction or being commanded to halt. The red arrow actually means "go" in the direction indicated is not perceived by the user to be the correct action as he believed that green indicated the direction to travel in due to the prior association.

7.3.2.3 USABILITY ISSUES

The field trial elicited comments regarding the limitations of the display screen, such as "the display size makes it hard to see where you are in the store once you move around for a while". Another comment was about the size of the text being too small to read. Users with visual impairments would have great difficulty in using the system. A positive comment about the application was that participants felt they could associate with the application better than using paper-based store guide (Scott, 2002).

The participants expressed enthusiasm for using the MapView in a store. However, they would still like the opportunity to ask staff questions and also retain the option of using traditional means of gathering information such as in store product information tags or advertisements. This echoes the finding we noted in the Virtual Notelet section. The users were enthusiastic to produce test messages, such as "test", "hello", etc. and try the infrared signal detection. The users, however, did not use the application after their initial attempt. Regarding the productview function, participants found it difficult to select a specific product from the screen with the product listings. Again, the text was too small in the map display.

A number of participants in the mega-store found the underlying concept useful for helping them find out whether something was in stock. A number of participants stated that they were satisfied with the dynamic nature of the map and that more information such as stock levels was provided than in the paper-based store guide. They concluded that the current paper store guide was described as "giving an overview" of the store rather than specific details.

Comments from managerial staff were obtained by holding an interview. The feedback shows the application gained enthusiasm and interest. The manager noted that it would be helpful and useful for customers. He also supposed that many customers would be interested in using it if such a system can really be implemented fully. However, he thought that having customers with unsupervised access to the devices raises concerns about the practicalities and security issues. A trolley mounted version may be more manageable in terms of these issues and workload for the members of staff (Asthana et al., 1994).

The soft-keyboard design in the MapView Shopping Assistant prototype did not follow generally accepted of HCI design principles, such as the discussion in (MacKenzie and Zhang, 1999). Our emphasis here is to validate whether other HCI designers can construct prototyping environment based on the GCS and software developers can build application using the GCS API. The user interface issue does not affect the validation of the GCS. Further development and improvement in the user interface could be carried out to address the current user interface issues.

7.3.2.4 SOCIAL ISSUES

On the subject of user observation, it was interesting to observe that the customers seemed happy to interact. Although they may simply have been unaware of this possibility and further work is required to explore this store assistant system.

From the manager's point of view, tracking customers' routes round the store as they interact within the shopping environment is useful in assessing if the store's merchandising policy is being maintained. It may highlight problem areas within the store and influence future shop layouts. More detailed, this information could then be cycled back into the store design in order to maximise the use of the available space and ensure that the current placement of goods maximises profitability. When customers use the application, it can log their movements around the store. Information from individual routes is then collated to a desktop PC and can be used for Customer Circulation Analysis.

7.4 SUMMARY OF THE MAPVIEW SHOPPING ASSISTANT

In this chapter, we have argued that:

- The case studies have shown that other programmers can use the API to build application based on the GCS environment.

- The evaluations highlight the difficulty of designing usable and useful context-sensitive location sensing systems.

The aim of this chapter is to assess whether the GCS environment can be implemented for large warehouse style retail stores. It is also used to address several specific problems, for example customers getting lost and not having enough information at hand. A wireless application could be built to provide customers and staff with mobile, location sensitive services to some of these issues.

Some feedback was positively received. For instance, users think that the existing store guide is useful for giving a high level overview, however, lacks specific details that context-sensitive mobile computing system can provide. Some features such as directions to a specific product cannot be found in a paper-based store guide. The results of the evaluation of the application also showed a myriad of problems that member of the general public face in using small screen PDAs. However, it must be stressed that our results are tentative and that more validation must be done. It is also important to stress that this case study was intended mainly to see whether other designers could use the GCSF API. This proved to be less problematic than the associated problems of finding a portable implementation environment for mobile applications.

CHAPTER 8
CONCLUSIONS AND FURTHER WORK

This thesis starts from the premise that context-sensitive mobile computing systems offer many potential benefits to future users of interactive systems. It is, however, difficult to validate the many claims that are made about these applications. Very few interface designers possess the necessary resources to build context-sensitive mobile computing systems. Previous approaches have relied upon specialist hardware or have involved re-wiring entire buildings for sensor deployment. We have presented an off-the-shelf approach to address these concerns. The hope is that the designers in this area can rapidly implement prototype systems and discover more about direct practical experience with this type of system. In this chapter, we first summarise our research. Secondly, we then highlight the contributions of this research. Thirdly, we present our thoughts on future work. More specifically, key directions we propose include a web application approach to tackle the issues we encountered during the development of the Glasgow Context Server (GCS). Finally, conclusions are presented.

8.1 SUMMARY OF THE THESIS

This thesis describes how we set out to engineer a prototyping environment for the development of context-sensitive mobile computing systems. This prototyping tool is intended to help HCI designers identify the human factors issues associated with a broad range of context-sensitive mobile computing systems. In the introductory chapter, we described the emergence of and need for context-sensitive mobile computing systems. In more detail, the advance in mobile computing, wireless communication, and sensor technologies creates a new form of human computer interaction. The burden of interaction between the user and mobile computing system can be reduced if the systems respond to change in the surrounding physical space. To achieve this, the computer systems need to know the user's tasks in certain situations and recognise potential actions. For instance, a context-sensitive mobile computing system might deduce that when a user approaches a particular artefact in a museum she is interested in the exhibit. The system might respond by displaying relevant information about this situation. Of course, such inferences may be incorrect. The user may not require the information at that moment. Hence, such potential pitfalls require the need for user feedback provided by prototype implementations.

Chapter 2 surveyed previous work in context-sensitive mobile computing. It categorised existing projects into different application domains and highlighted the claimed benefits for these types of systems. We concluded the most significant benefit of a context-sensitive mobile computing system is that it can help reduce the explicit input required from the user. Hierarchical Task Analysis (HTA) was exploited to analyse each application domain. The intention was to validate the claimed benefits and show how a designer might find the context information used by mobile computing systems.

Chapter 3 investigated a number of context sensing infrastructure for the projects discussed in the chapter two. The context acquisition techniques focus on the user's location and surroundings that are considered relevant to their current task in the immediate situation. In addition, this chapter pointed out the limitation of the existing approach to context-sensitive location and environment sensing. This formed the basis for the development of an off-the-shelf approach in this thesis.

Following this, Chapter 4 described the design of a prototyping environment for context-sensitive mobile computing. This was motivated by the issues raised during the implementation of previous approaches to location and environment sensing described in the chapter three. Our design was developed to enable the HCI designers to rapidly build context-sensitive mobile computing applications using off-the-shelf hardware and software technologies. The designers can use the applications built upon the prototyping architecture as test-beds to discover more about human computer interaction issues in this new form of computing.

Chapter 5 described the technical challenges of implementing such a prototyping environment. In particular, this chapter focused on the infrared sensing technique and the API for the designers to develop applications based on our prototyping architecture. Two primary reasons motivate this presentation. Firstly, it illustrated the sorts of obstacles that designers might encounter and issues that they must take into account if they want to follow similar approaches to context-sensitive mobile computing. Secondly, the aim is to share our technical experience in infrared sensing in the hope that this might act as a blue-print for other HCI designers.

A web-based context-sensitive mobile computing annotation system, the Virtual Notelet, built upon the GCS was presented in Chapter 6. The aim of this chapter is to demonstrate the design process mentioned in Appendix B summarised in the following paragraph. We also show how the GCS environment can be used to build a working system. In addition, the Virtual Notelet application was exploited as a test-bed for gaining end-user experiences in using such a system. Similarly, a context-sensitive shopping assistant, the MapView, application also built on the GCS was presented in Chapter 7. The purpose of this chapter is not only demonstrating the GCS application but also to show that other programmers can use the API described in Chapter 5 to build a context-sensitive mobile computing application. The remainder of this chapter describes the major contributions of this thesis and addresses potential future work before presenting concluding remarks.

A design process for context-sensitive mobile computing was described and discussed in Appendix B. In particular, this chapter concentrated on how to identify useful context information in the system design. Computer applications help their users accomplish tasks in their particular work domain. A number of previous researchers have argued that a context-sensitive mobile computing application can provide its user relevant information based on their current situation with minimal user input requirement by exploiting "context" (Lieberman and Selker, 2000, Selker and Burleson, 2000). Our work focuses on three essential questions when designing a context-sensitive mobile computing application. Firstly, what is the context information? Secondly, where to find the context information?

Thirdly, how to use the context information? However, this is not yet a fully articulated design method. Further development continues.

8.2 CONTRIBUTIONS OF THE THESIS

The major contributions of this thesis are:

- The development of a prototyping environment, the GCS, for context-sensitive mobile computing, plus insights into architectural issues.
- Task oriented approach to identification of function allocation in context-sensitive systems.
- Demonstration of two working applications built upon the GCS and insights into usability problems of these applications.

The following sections describe each of these contributions in more detail.

8.2.1 GLASGOW CONTEXT SERVER (GCS)

The development of context-sensitive mobile computing systems requires considerable engineering skills. This could be an obstacle to the HCI designers who want to discover more about detailed social or usability issues but do not possess knowledge about the hardware or software design and implementation for this type of systems. The GCS can help this type of HCI designers to implement a prototyping environment without recourse to the specialist hardware of previous approaches. As for the development of the applications based on the GCS API, the users (software developers) of this API can be categorized into library and toolkit programmers. The former simply exploits the existing support whereas the later will alter and add APIs that have not existed to suit their needs in the application development.

The GCS and the GCSF API, which helps the designers to develop applications based upon the GCS, is built from off-the-shelf hardware and software. The aim is to minimise the costs associated with installing the system and simplify its maintenance. Our work built upon the Active Badge research of AT&T Research, Cambridge and Xerox by utilizing developments in mobile computing devices and wireless communications. The initial net effect is to enable users to keep track of their location without the need to wire-up an entire building. This avoids the significant installation and maintenance overheads that have limited the application of previous approaches. The current implementation of the GCS exploits domestic infrared remote controls to provide low-powered diffuse

infrared transmitters. Depending on the type of interaction that is required, these infrared transmitters can either be used as beacons to indicate particular locations in a building or as tags to help users obtain information about particular objects in their environments. It is worth noting, however, that the GCS has a modular design that is consistent with both the Bluetooth standard and with differential radio signal detection. These more elegant approaches can simply be substituted for the infrared beacons that we currently use as a pragmatic alternative. The design principles behind this architecture were that it should be low cost, should be modular and easily extensible and that, above all, it should exploit off-the-shelf hardware and software. The GCS and its API, the GCSF satisfies these requirements.

This off-the-shelf approach to hardware design has also been carried over into the software components. The GCS, therefore, makes extensive use of the web-based HTTP protocol and Java Servlets. There are some similarities between the GCS and Hewlett Packard's cooltown proposals, which provide a vision that human and inanimate objects are interconnected over the web. This approach offers numerous benefits for the development of context-sensitive mobile computing interfaces:

Ubiquitous and Heterogeneous Platform Support

The web infrastructure is ubiquitous. Ideally, both servers and browsers will execute on various types of devices. This is essential for an off-the-shelf approach to context-sensitive mobile computing interaction. Adopting this approach, it should be possible to integrate users' existing PDA's into the GCS environment. The GCS system should also avoid making strong assumptions about the hardware characteristics of the server. The web communications infrastructures are also ubiquitous. These range from traditional Ethernet networks through to radio frequency LANs and cellular architectures using the Wireless Access Protocol (WAP) and similar off-shoots from the HTTP.

Flexible Information Caching Support

As we noted in Chapter 3, previous approaches to context-sensitive mobile computing lacked off-line support for their users. The HTTP provides caching mechanisms that can be used to provide the on-line and off-line services that can have a profound impact upon the usability of some context-sensitive mobile computing application. For instance, servers typically implement a HEAD operation that returns information about the status of any web page. This can include an estimate of the time that it is safe to hold any page in a cache before it is likely to be updated. As we described in Chapter 4, the designers of mobile computing applications can use this data to reduce the amount of information that may have to be transferred when a user moves into and out of radio frequency LAN coverage. Of course, the off-line support of HTTP is far from ideal. This remains a focus of continuing research.

8.2.2 TASK ANALYSIS FOR CONTEXT-SENSITIVE SYSTEMS

HTA is used to identify context information from the task and interaction flows in existing context-sensitive systems as well as the Virtual Notelet and the MapView shopping assistant application. Our research shows the strength and weakness of this technique. In many places it shows helpful to us in identifying context information when designing the Virtual Notelet application. However, prototyping experiences have also showed that the HTA cannot guarantee a useful product. For example, it does not capture social issues very well, enjoyment in playing a context-sensitive game, etc. Further work may need to be done to produce a better technique on these issues.

8.2.3 GCS APPLICATIONS

As mentioned in previous paragraphs, there are some similarities between the GCS and the cooltown project. Although we have exploited the GCS to implement some of these ideas, we would follow a less ambitious path. The intention is not to revolutionise current tasks by enabling inanimate objects to sense their location. These applications may offer significant long-term benefits. In the short term, however, we believe that HCI design techniques must be exploited to identify those existing real-world tasks that benefit the most from context-sensitive mobile computing. Our initial approach of using the built-in infrared communication support for existing mobile computing devices has enabled us to build prototype systems that can demonstrate the benefits of context-sensitive mobile computing systems in a range of existing environments ranging from offices to shopping malls.

The Virtual Notelet was implemented to simulate a context board in the office domain. The user can observe the office occupant's current status and decide whether it is suitable to pay a visit or leave a note. The users can retrieve the occupant's status as a virtual note associated on a physical office using their mobile computing device. They simply point their mobile computing device at an infrared transmitter and the relevant virtual notes are then displayed on their device. We took this working application as a test-bed to evaluate how people conceive this type of application by conducting a field trail.

The results of the Virtual Notelet field trail showed that the users exploited a host of more conventional communications media such as physical notes and electronic mail. They had developed expertise in using these applications to support their daily tasks. It is, therefore, difficult to persuade them to continue using the Virtual Notelet application. However, this does not mean that context-sensitive mobile computing systems offer few benefits to their users. Our findings do, however, confirm the importance of obtaining direct evidence from field trials before making more elaborate claims about the potential benefits of context-sensitive mobile computing. The field trial of the Virtual Notelet application showed that task analysis is insufficient. One must do prototyping and testing. A preliminary conclusion from this work is that context-sensitive mobile computing systems not only

depend on a revolution in their underlying technology but also require fundamental changes in the way in which people access information as they move around their physical environment.

The MapView shopping assistant can help its user to keep tracking of their current location within a store. This application can also provide information about the products nearby. It can guide the user from the current location to the location of a particular product she wants to buy in the store. This project was developed by two final year students using the GCSF. This application was functioning and exploited to conduct a field trail in the mega-store. During the development, there was a significant issue about the context information presentation. The map, which displays the user's location, cannot guarantee the user's orientation. The infrared sensing used in the GCS cannot work out which direction the user is positioning in. The infrared signal simply provides as an indexes of locations or products in the store. In addition, development problems came from the device of implementation environment on the target mobile computing device. We realised that providing a GCS style platform does not solve all of the problems in building context-sensitive mobile computing applications. The meta-level message here is that even Java style virtual machines may not be as portable as we thought.

As for the field trails, we received some positive feedback. For instance, users think that the existing store guide is useful for giving a high level overview, however, it is lacking in more specific details that the context-sensitive mobile computing system can provide. Some features such as directions to a specific product cannot easily be found in a paper-based store guide. The store management were also interested in exploiting customer tracking features.

8.3 FUTURE WORK

We began our research with the intention that we would focus on the human factors issues in context-sensitive mobile computing systems. In particular, we were anxious to validate the claims that previous authors have made about the potential benefits of these systems. In order to do this we were forced to develop a test-bed for conducting usability studies in this area. The GCS's integration of off-the-shelf hardware with elements of the existing web infrastructure partially satisfies this requirement. It is not, however, a panacea. Development work continues.

A number of potential problems arose when using the web to support context-sensitive mobile computing interaction. Some of these problems have been recognised in previous research, such as the task of exploiting off-line support and the need for more fine grained access control mechanisms (Kindberg et al., 2000). Other limitations emerged through the development and evaluation of the Virtual Notelet and the MapView case studies that have been described in Chapter 6 and 7:

Browser Modification

Minor modifications must be made to the browsers that run on end-users' PDAs. The typical web model is one in which users' initiate the transfer of information by selecting hyperlinks or by entering URLs. In contrast, context-sensitive mobile computing systems may initiate the transfer and presentation of information in response to the detection of a location beacon or an object tag. In order to make these modifications, it is typically necessary to have access to the browser's source code. This limits some of the claims made in the previous paragraphs. Although it is possible to access the web through a vast array of devices, it can be difficult to implement the small number of changes that are necessary before a browser can be completely integrated into the GCS environment.

Log of Content Request

The web-caching model does not provide adequate support for location-sensitive interaction. For instance, if an HTTP connection cannot be established then most browsers will simply return an error code. In contrast, the GCS system must log such failed requests. It may be possible to support user tasks by repeating the request once they return to radio coverage. These logs can also be used to improve predictions about the useful content of a cache during future interaction.

Fine-Grained Access Control

Existing Web protocols do not provide the fine-grained access control mechanisms that are necessary for mobile computing interaction in a way. It is necessary to determine whether or not a user has permission to access information or to request services from the web-enabled objects that are an important feature of HP's cooltown proposals. Similarly, our Virtual Notelet case study shows that even for location-disclosing systems, the ubiquitous nature of web access makes it critical that privacy is considered and maintained. It is possible to implement access control policies using the existing web infrastructure. For instance, `.htacces` files can be placed in restricted access (Stein and Stewart, 2001). This approach is error prone because users often forget to remove access restrictions or leave sensitive information unprotected. Alternatively, server administrators can alter the directory access control settings in global files, such as `access.conf`. This approach is inflexible. Individual users cannot easily alter the permissions that are associated with their files. The fact that we must consider such details provides an ample illustration of the difficulties posed by using the web as a principle component of the GCS. Fortunately, it is possible to hide much of this complexity by providing appropriate abstractions to interface designers. It is, however, less easy to identify abstractions that end-users can exploit to dynamically alter the permissions associated with their location sensitive information. This remains a subject of current research.

It is important to place these limitations in the wider context of recent attempts to revise the infrastructure that supports the web. For example, improved access control and cache monitoring have

a wider significance beyond the implementation of context-sensitive mobile computing interfaces. Such techniques have the potential to improve the design and implementation of many different interactive, web-based systems. Attempts to develop the next generation of HTTP, HTTP-NG have described the existing mechanisms for controlling the browser cache as not precise. W3C working groups have, therefore, begun to identify means of improving the performance that is implied by new generations of mobile users with sporadic connections (Stein and Stewart, 2001). These changes will have a significant impact upon the usability of the web in general. By integrating HTTP and successor protocols into the GCS, we are also in a position to benefit from these developments.

In Chapter 5, we presented two different approaches to implement applications based on the GCS. The initial focus was on interfacing the mobile computing device and infrared sensor and created GUI widget-like sensor component. The resulting effect is that the designer can incorporate sensors, for example, the infrared receiver, in an application by "drawing" them on the application. The second approach exploited Java Servlets to implement computation on the server side. A servlet can be considered as an applet that runs on the server side without a GUI. This approach addressed hardware limitations and minimal programming toolkits for mobile computing devices. As stated in Java's official website, Servlets can access the entire family of Java APIs. These include the APIs to access enterprise databases, a library of HTTP-specific calls, and etc. The designers simply focus on implementing the user interface and sensor interface using GUI widgets and sensor components on the target mobile computing device. The internal computation of the application is implemented on the server side. Communication between the user and the sensor interface on the mobile computing device and the correspondent internal computation is through the HTTP protocol.

As we noted in the Chapter 7, implementation language incompatibility issues frustrated this development of the MapView context-sensitive mobile computing application using the GCSF. The incompatibility, in particular, affected the development of the graphical user interface. The GCSF mainly focuses on sensor and communication development. It was implemented using classic Waba. However, Waba provides a restricted number of user interface components. In order to access more GUI components to suit the MapView shopping assistant project, they exploited SuperWaba, which is based upon classic Waba, but has more GUI components. However, some GUI components that are provided by SuperWaba are not available for our target devices.

We propose a web application approach. Web applications are those software tools and utilities that run on a server and interface through a web browser. Leaving the computation power on the server side and presenting the information through a web browser can free the designers from the incompatibility issue we addressed. However, the ability for a web browser to respond for implicit input from sensors is still the main concern of this approach. A context-sensitive mobile computing system may initiate the transfer and presentation of information in response to the detection of a location beacon or an object tag. In order to make these modifications, it is necessary to have access to the web browser's source code. This limits some of the claims made in the previous paragraphs. We,

therefore, exploit the web proxy approach to build an intermediate proxy between the web browser and sensor components.

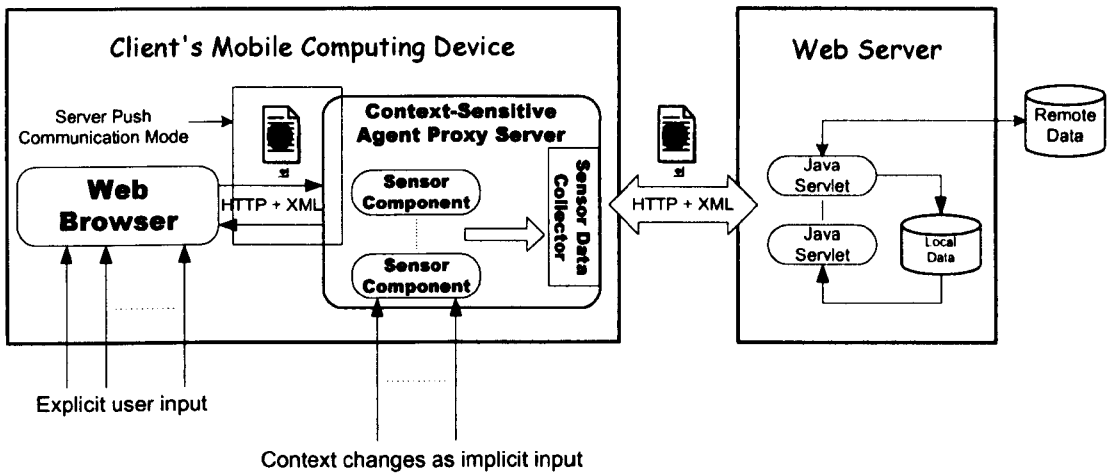


Figure 8.1 Context-Sensitive Agent Proxy Server

Figure 8.1 presents a revised architecture for the GCS that has emerged from this thesis. The Context-Sensitive Agent Proxy (CSAP) Server acts as a web server to the web browser that runs on a mobile computing device. The designers can draw sensor components within the proxy server in the same way as we built the Virtual Notelet and MapView application described in Chapter 6 and Chapter 7 using the GCSF. The Sensor Data Collector gathers the sensed information and initiates a HTTP request with sensed data embedded to the corresponding servlet. It also acts as an arbiter of sensor components to decide what sensor data are needed for executing a particular application. The results are then pushed to the web browser. From the web operation perspective, there are two methods of communication between a web server and a client. Normally, a web browser follows the client pull approach, which initiates events that send requests to the web server for the web pages. Under some circumstances, it is necessary to have the server send data to its clients. For instance, a web server that hosts stock information has to initiate communication events to its clients in order to remain in synchronisation with up-to-date stock data even if the clients do not request the data per se. This is called server push. In the case of CSAP, up-to-date results must be pushed to the web browser without the users firing explicit requests. The idea is that once the first chunk of data is transmitted from the web server, the client then waits for the server to send further data. The web server has to keep the network connection socket between the server and its clients open. This approach can cause the issue of server overload. Our proposed architecture addresses the problem and has the proxy server reside on the client side. There is only one connection remaining between the browser and the CSAP server. This is a preliminary sketch and much work remains to be done to establish whether this architecture might provide significant benefits over the existing GCS approaches.

8.4 CONCLUSIONS

The aim of this research was to provide a means of exploring whether or not some of the claimed benefits for context-sensitive mobile computing systems could actually be realised by the evaluation of prototype systems. We, therefore, developed the GCS, which is designed to provide a low-cost test-bed for the use of context-sensitive mobile computing in human-computer interfaces within an indoor environment. The GCS provides an extensible architecture for designers in the field to build context-sensitive mobile computing systems using off-the-shelf hardware and software components. The use of low-cost techniques was born from pragmatic constraints. We had limited software development resources and wanted to entirely avoid any reliance on specialist hardware. In addition, we wanted to avoid any changes to the infrastructure or fabric of the buildings that would host the system. GCS applications aim to be simple to maintain and future proof. It is possible to update system components when new technologies become available. In contrast, previous work in this area has relied upon additional specialist hardware and software to build context-sensitive mobile computing application. This exacerbates the development of this type of application. Previous research efforts have focused on developing innovative hardware and software support for this new form of interactive system (Dey, 2000, Hong and Landay, 2001, Kindberg et al., 2000). Our research effort focuses more on supporting the development of prototype context-sensitive mobile computing systems. The ultimate hope is to help researchers validate the usability and social issues associated with this new form of interactive system.

APPENDIX A
DESIGN GUIDELINES OF CONTEXT-SENSITIVE MOBILE APPLICATIONS FOR NOVICE USERS

We must know who the application is designed for in order to make the application support users' needs. This can be done by classifying users according to the various characteristics they might have in common. Context-sensitive mobile computing has never been widely available to users in their everyday life {Guanling Chen, 2000 #9}. In this thesis, we exploited rapid prototyping approach to suit this situation as it is argued to act as a means of training the users to a specific standard prior to the introduction of the final product when the form of a system is uncertain (Dix et al., 1998). We must also tend to design this type of applications to provide novice user support. Also we cannot count on user's previous experience with desktop applications.

Interaction between a user and a context-sensitive mobile application is different from the conventional mouse-keyboard systems. Explicit user input may become implicit because the applications can take advantage of context information using sensors. Users of this type of applications can be considered as novices in terms of the transformation of explicit to implicit interaction. In this design process, we focused on the way that users interact with this type of application and how the applications obtain and exploit the implicit input.

Faulkner's concerns are particularly relevant here because very few people have ever used a computer application that exploits the richer forms of context sensitivity hypothesised by Dey et al. Although, as we shall see, the potential users of these systems may be experts in the use of more general applications, it is appropriate to consider the additional support required by novice users during the development of this new generation of interactive systems. We apply the design principle of applications for novice users remarked by Faulkner to the design of context-sensitive mobile computing systems are described in the following:

A. All initiatives should come from the computer.

   The aim of a context-sensitive mobile computing application is to take as many chances as it can to obtain context information about its user's current situation while the interaction between the application and the user is carrying on. Again, the intention is to reduce explicit user input. This is important to minimize the distraction while the user is performing her tasks. Context-sensitive applications may obtain implicit input from unconventional devices, such as GPS receivers, RFID tags, tilt sensors, or infrared sensors. We categorize the interaction between a user and sensors into *explicit interaction* and *implicit interaction*. Applications that exploit sensors in obtaining context information pose two different types of challenges for human-computer interaction. Firstly, some applications may still need their

users to perform additional tasks to interact with sensors in order for the applications to detect and translate the sensor data into suitable context information. For example, when exploiting Java iButtons to provide user, location, or physical object identifications as context information, the user is required to perform an explicit task. They must dock the iButtons to its reader, which can be installed on a computing device, at a physical place. Secondly, applications can obtain context information from their users' natural activities such as walking, gesturing, and other real-world physical interaction. The application's context acquisition process is "embedded" in their users' acts when they perform tasks. For example, the user's current location can be detected by the application acquiring location data from a GPS receiver while the user is walking. Designer of this type of system must provide description and instruction about how to interact with the application using the mobile device once the application starts.

B. Each necessary user input should be simple and brief.

The less requirement of user input reduces the necessary teaching for the user to accomplish tasks and makes the human-computer interaction easier (Selker and Burleson, 2000). As noted in the above paragraph, context-sensitive mobile computing applications exploit sensors to obtain context information. The users may be asked to perform tasks to fulfil the sensing process or simply focus on their primary task without the distractions from the sensing process. The amount of explicit user input should be reduced in both cases.

C. Each user input procedure should be consistent with user expectation

Application designers expect that their assumption of why, when, what, and how to use the application would meet their users' expectations. When designing a context-sensitive mobile computing application we should present the user interface in a way that match as the user 's expectation of the systems. However, with a new system, the users' expectations are likely to be inaccurate. Again, this is because that they will not have used a system like it before hence they will not know quite what to expect or they may expect things that are not possible given current technology. To tackle this, the proper metaphors that the users are familiar with should be applied in the design process (Coschurba et al., 2001).

D. No necessary special training

For novices, an application should apparently provide information that is needed to operate the application. Again, exploiting metaphors that the users are familiar with can reduce the requirement on the users. Designers should know the behaviour of their users when they perform acts to accomplish a task with an application. If the user is required to perform explicit interaction with sensors in order for the application to detect and use the sensed data as context information, the interaction should be close to the activities that the user expects

based on her daily experiences. For instance, the In/Out Board application at GVU utilized Java iButtons to provide user, location, time identity. The user is required to dock her iButtons to the reader attached at the entrance of her office when entering or leaving the office. The act of docking the iButton on the reader is similar to the non-computer support alternative that most people use in the office domain. In the GCS, the user is required to put her mobile computing device close to infrared transmitters.

E. All system messages should be clear and unequivocal

All messages to the user should be obvious. Lengthy descriptions or system messages displayed on the screen of a mobile device would make the users feel difficult to read. This implies the need for user testing and hence the importance of prototyping environments likes the GCS. User test can help discover questions like how will we know what is clear given that there may be errors that emerge from new forms of interaction with this type of systems.

F. User decisions should be made from a small set of options

Applications should provide small choice set to the novice users when they need to pick up one or more of the choices as input to the applications to carry on the operation. Providing ease of use and reducing the teaching needed for the user to accomplish tasks is the aim of context-sensitive mobile computing applications (Schmidt, 2000, Selker and Burleson, 2000, Tuulari, 2000). Therefore, in particular, the explicit user-sensor interaction should be prompt and suit the users' expectation about performing the tasks.

G. Users should control the pace of interaction.

Novice users prefer the feeling of control along the interaction with an application. From the application's point of view, it might bring convenience by filtering out unimportant things taking place at the moment for the user, but the novice user likes to be in control (Faulkner, 1998). This evokes the trade-off between explicit and implicit user-sensor interaction. In the case of implicit user-sensor interaction, the application obtains context information from the sensors that recognize contextual information through the user's acts without asking her to do so. The way of automation may bring convenience to the user and she can focus on her primary task without the distraction from the user-sensor interaction. On the contrary, applications that adapt explicit user-sensor interaction provide their users the feeling of control when the users perform tasks with the applications. A number of authors argued that there is controversy about privacy over the explicit and implicit user-sensor interaction. Applications that exploit implicit interaction may disclose the sensed context information in order to fulfil system tasks while the user is not willing to do. In such a case, an appropriate level of trust in the application must be provided to the user.

H. User decision making should be in response to a specific request for action.

Users should not be expected to guess the appropriate time to perform their tasks with applications. The designer of applications that exploit explicit user-sensor interaction should know when the interaction needs to be done and get the application to prompt if something needs to happen. Explicit interaction should be as close to the activities that the users already perform to achieve a certain goal based on their previous experiences. Implicit interaction can avoid the hesitation that the users may encounter in the use of explicit user-sensor interaction.

I. There should be sufficient feedback.

Applications should notify their users about the progress of the interaction. It is important for novice users to know whether they are on the right track. This can be done in a number of ways. For example, the response can be graphical or textual in a GUI application or textual in a command-line application. The idea of graphical and textual response can be channelled to the feedback of sensing progress from a context-sensitive mobile computing application. Sensing processes, both the *explicit and implicit interaction* should provide notification to the user once closure occurs. This is particular important for the explicit interaction. The user likes to know if she performs the correct acts to trigger the sensing processes. The notification of successful or failed sensing process can be graphical or textual output on the screen of the user's mobile computing device. As noted by other researchers, mobile computing devices would cause distractions and make them difficult use while the user is doing something else (Anhalt et al., 2001, Cheverst et al., 2002, Tuulari, 2000). This is particularly due to the size of screen and keyboard is too small to operate. In the GCS, we adapt audio output from the device once the sensing processes success. The distraction from user-sensor interaction is minimized. However, a challenging issue in this case is that when to minimise the context-sensitive information. As an example, if the system is unsure of the users location what should happen then? Should the system explicitly prompt the user or should it just go back to making no assumptions about their location?

APPENDIX B
DESIGN PROCESS FOR CONTEXT-SENSITIVE MOBILE COMPUTING APPLICATIONS

Computer applications help their users accomplish tasks in their particular work domain. A number of previous researches have argued that context-sensitive mobile computing applications can provide users with relevant information based on their current situation with minimal user input (Lieberman and Selker, 2000). There are three essential questions when designing a context-sensitive mobile computing application. Firstly, what is the context information? Secondly, where to find the context information? Thirdly, how to use the context information? However, producing a suitable design process for a context-sensitive mobile computing application is a non-trivial task due to the considerable disagreement over the definition of "context".

According to a number of dictionaries, the definition of context is influences and events that helped cause a particular event or situation to happen (*Cambridge International Dictionary of English*). In writing, context is words that come before and after a word, phrase, statement, etc, helping to show what its meaning is (Manser, 2000). While these are general definitions it does help for understanding the preliminary concept of context-sensitive computing systems. In many context-sensitive mobile computing applications, context is also viewed as the input and output that the user received up to the current point in interaction. The influences and events can be considered as input that is expected by a certain computer application or system to process and generate the result relevant to the happening situation. As we noted in Chapter 1, Dey et al gave the definition of context as a piece of information that can be utilized to describe the situation of a participant in an interaction. Their definition of context is rather close to the meaning of context in writing. A participant in an interaction can be explained as a user performs tasks to achieve a certain goal with a system. The answer to the question, what the context is, seems clearer if we fuse the definitions from dictionaries and Dey et al. The context information for an application can be considered as the inputs that are required to perform the tasks that the application intends to perform to help its user achieve a certain goal in a situation. The input mentioned here is referred to implicit input that is not intentionally performed by the user to operate a system but conceived as input by the system (Schmidt, 2000). For instance, a user carrying a handheld computer is interested in the artefact in front of her in a museum resulting in the information about the artefact displayed on the computer. The user performs the task, "walk to the artefact interests her", is regarded as an input to the system to perform system tasks to suit the user's current situation.

Context information also determines the situation that the application acts on. In order to provide an operational method to identify the context information when designing a context-sensitive mobile computing application using the GCS environment, our approach is to exploit scenario-based design

method to picture user activities in given situations and analyse how context-sensitive mobile computing applications could be used to reshape user activities by looking at the fundamental design framework in HCI based on the scenarios (Carroll, 2000b):

1. User model
2. Task analysis
3. Operating environment

Scenario-based design approach helps us concentrate on how users accomplish their tasks and keep the focus on situations of users' activities (Carroll, 2000a). The benefit of doing this is to identify the context of users' activities when they perform tasks to accomplish a certain goal. It is important for the designer to determine which kind of users' activities can be transformed to computer applications by increasing the communication channels between human and computer applications. As we noted in Chapter 2, the HTA was used to identify what user task can be transferred to systems task based upon specific context. The following describes how we consider the user model, task analysis, and operating environment in a context-sensitive mobile computing environment, in particular, the GCS operating environment. The intention of this is to ease the task of design this new type of applications using the GCS by analyzing these fundamental questions in given scenarios. To suit the situation of designing a context-sensitive mobile computing application, we revise these questions as follows:

1. How to describe the user → What is the role of a user in a scenario?
2. What is the Task? → Where to find the context?
3. What is the environment where the application into play? → How to obtain context information from context?

The answers to these questions are described as follows:

B.1 USER MODEL: HOW TO DESCRIBE THE USER?

We must know who the application is designed for in order to make the application support users' needs. This can be done by classifying users according to the various characteristics they might have in common. Context-sensitive mobile computing has never been widely available to users in their everyday life. In our research, we exploited rapid prototyping approach to suit this situation as it is argued to act as a means of training the users to a specific standard prior to the introduction of the final product when the form of a system is uncertain (Dix et al., 1998). We must also tend to design this type of applications to provide novice user support. Also we cannot count on user's previous experience with desktop applications. Interaction between the user and the applications is different from the conventional mouse-keyboard systems. Explicit user input may become implicit because the

applications can take advantage of context information using sensors. Users of this type of applications can be considered as novices in terms of the transformation of explicit to implicit interaction. In this design process, we focused on the way that users interact with this type of application and how the applications obtain and exploit the implicit input. Faulkner's concerns are particularly relevant here because very few people have ever used a computer application that exploits the richer forms of context sensitivity hypothesised by Dey et al. Although, as we shall see, the potential users of these systems may be experts in the use of more general applications, it is appropriate to consider the additional support required by novice users during the development of this new generation of interactive systems. We apply the design principle of applications for novice users remarked by Faulkner to the design of context-sensitive mobile computing systems are described in Appendix A.

The introduction and the use of wider social information in computer applications is a research area in its own right. In anticipation of the results of this research, the remaining chapters of this thesis will focus on a relatively simplistic representation of an individual's role. This will capture details such as their name, job description or activity such as "customer" or "sales adviser", primary tasks and so on. The task-relevant background, for instance, social aspect, should be taken into account in the user model. It affects how we describe what a user is doing in a certain context of a situation. A user can play different roles in different situations. A role can be characterised with, for example, a use's name, status, tasks that a user can perform in a specific role (Myrhaug, 2001).

B.2 TASK ANALYSIS: WHERE TO FIND THE CONTEXT?

When designing an application, the first step is to understand what the application should be doing. What an application can do to help its user achieve a certain goal can be characterised with the interaction between the application and the user. Task analysis is exploited to understand what needs to be accomplished by the application and break down the major task, the purpose of the application, into the simplest component part. Each component consists of clear goals, tasks, and actions. In order to carry out the application we need to know what information is needed for each task. The information may come from the user input or the state of current runtime environment. Traditional human-computer interaction requires explicit user input whereas context-sensitive mobile computing applications can adapt both explicit and implicit input from context information, for example, spatial position, pattern of motion, and etc (Masui and Siio, 2001, Selker and Burleson, 2000). In the case of designing a context-sensitive mobile computing application, we need to know what user tasks are necessary to operate the mobile computing device and also need to figure out which part of user input that is necessary for the application to perform the tasks can be transferred to the application to obtain the information by itself in order to increase the application's level of context-sensitive. On the other hand, the application can adapt to its user's information need with minimal explicit user input by taking advantage of context information.

Our approach is to extract user tasks from situations that are elicited from a scenario. As noted by Carrol, scenarios are stories about people and their activities (Carroll, 2000a, Carroll, 2000b, Carroll and Rosson). Each scenario has a setting that explicitly describes the starting state of the current situation and implicitly depicts the characters that take part in the situation in the scenario. Each scenario has actors who perform tasks to achieve goals in different situations in a scenario. Each task can be regarded as what needs to be done in the situation. We analyze the user tasks in terms of the answers to the questions, "Who should be responsible for the situation?", and "What should be known to act on the situation?". The Hierarchical Task Analysis (HTA) is utilized to picture and describe how a situation happens and user/computer application tasks performed in a scenario. In order to figure out the transformation between user and computer application tasks, we adapt the Entity-relation-based analysis technique to identify the relationships between entities, actors, and actions described in the HTA. In addition, the user would feel easier to stay in role and resolve any potential hesitation if the adapted scenario can reflect situation based on the user's previous experiences with realistic reasons for performing the tasks. The closer that the scenario represents reality, the more chance the useful context is discovered.

As mentioned previously, we are interested in user tasks in a situation within a scenario. In particular, we focus on the actors, goals, and settings (situation) of a scenario. We concentrate on the way users perform tasks to accomplish goals. The point is that task analysis can help us to move from a scenario to a more concrete design. User testing can then be used to observe limitations with the task analysis that will only be apparent when real people actually start to use the system. We exploit Hierarchical task analysis (HTA) to describe the tasks and subtasks that are performed by users in order to achieve goals in situations. In addition, the plans in HTA help describe the condition and order that the tasks should be performed. We also utilize Entity-relation-based analysis to identify the relationship among actors, objects, and actions when users perform their tasks. The aim is to figure out which part of user tasks can be transformed to computer applications if the level of context-sensitive is increased on the computer application side. The task analysis techniques are applied to each scenario in order to identify situations, actors, goals, and actions that include both with and without computer application support when users perform tasks in each scenario.

B.3 OPERATING ENVIRONMENT: HOW TO OBTAIN CONTEXT INFORMATION?

Sensors, both hardware and software implementations, are utilized to increase the communication channel between users and context-sensitive computer systems. The sensed data can be modelled in terms of the operating environment, user, and user-computer task. This can widen the common knowledge between users and computers and make the interaction appropriately. As we mentioned in Chapter 1, an Augmented Reality system provides its users a synthesized world on top of the real world through a see-through 3D display. In the case of the GCS environment, users carry wireless-

enabled mobile computing devices while they are moving around physical spaces. The mobile computing device can detect its current location and surrounding physical objects using infrared sensing technology. The applications based upon the GCS environment can exploit and act on the sensed information to provide their users with tailored information that reflects their current situation.

We would like to emphasis that this type of interaction is similar to Augmented Reality. The wireless-enabled mobile computing device can be considered as a bridge between its user and their physical space. A number of authors step further to the more specific paradigm of Augmented Reality and Mobile Computing, others focus on Personal Augmented Reality, which utilizes wearable computers with personal display devices to augment a user's perception of their current physical space (MacIntyre, 2001). From the system architecture point of view, the fundamental context information in this type of system is location and surrounding physical entities, including people and objects such as printers, projectors, etc.

As we noted in Chapter 3, Tuulari et al argued that it is important to divide physical space into natural and artificial parts (Tuulari, 2000). It helps interface designers better understand what, where, and how to obtain the context information in the operation of this type of system. In some cases, the users can obtain information about their surroundings, both natural and artificial parts, or other sources of context easier using artificial sensors than using natural sensors. Some examples that are illustrated to support this argument are: measure temperature (natural part) using thermometer (artificial part) and satellites and GPS devices (artificial parts) are exploited to measure the parameter of the user's location (nature part). In the current GCS implementation operating environment, the architecture (artificial part) can transfer information from both the nature part (location, nearby people) and artificial part (computing resources) to the user.

B.4 ACTION: HOW TO USE CONTEXT TO TRIGGER PERFORMANCE OF CONTEXT-SENSITIVE BEHAVIOR

An application is designed for a particular purpose. The designer of a particular application must have some idea of the circumstances where an application can be used. In context-sensitive systems designers decide what actions should be activated when a specific situation happen. The context of the situation, both explicitly and implicitly comes from the users, the surrounding environment, the system architecture itself, or anything related to the situation is utilized as inputs to the application. In previous section, we mentioned that our approach of converting explicit user input into implicit input from sensing mechanisms helps to transfer the effort of user's task to the computer application. Once the tasks, which the user is expected to perform in a situation are determined, designers can evaluate which user tasks can be performed by computer applications through the use of appropriate sensors.

B.5 SUMMARY OF THE DESIGN PROCESS

The GCS provides designers with a prototyping environment as a test-bed for context-sensitive mobile interaction. From the technical point of view, the prototyping environment simply simulates some, but not all, features of the intended system. This is due to the fact that more and more cost-effective sensors are available for context-sensitive mobile computing system development (Abowd et al., 2002, Dey, 2000, Tuulari, 2000). There are possibilities to adapt new sensors in the development of context-sensitive mobile computing application. As we mentioned in Chapter 5, the GCS provides a flexible development environment. We follow an incremental prototyping approach to build new types of applications based upon the GCS (Dix et al., 1998). This approach helps designers to focus on one overall design for the final system. The designers build independent components and add to the system until it is completed.

The concept of software design is to envision and facilitate new method of dealing with things and new things to do using computing technology. Humans' experiences and activities illustrate detailed constraint for the development of computer applications. The designers, therefore, must have better understanding about how computer applications can be transformed or limited by the context information about the user's activities of doing things, when they carry out the design process. We adapt scenario-based design approach for a number of reasons. It helps designers envision and log the user's activities from the beginning and along the development process. Also, it helps to reason about how situations, users, and user's activities happened in a scenario. Most important, it can help the designers focus on the assumptions about users and tasks that are implicit in application and systems. A scenario consists of sequences of actions and events. Those are answers to the questions: What is the user doing? What is happening to the user? What are changes in the current situation? Each scenario representation can be detailed as prototypes using rapid prototyping utilities such as the GCS.

To summarise, the design process starts from setup a scenario, identify the user, break down the scenario into situations and identify triggers for the situations (context), identify what needs to be done in each situation in the scenario using task analysis. The next step is to embody context (decomposition of the triggers) information as input to the application from each task by identifying what needs to be known to perform the task, distribute context to the user and the application as input to their tasks and perform actions should act on the situation. Figure B.1 describes how we consider the position of context in a scenario.

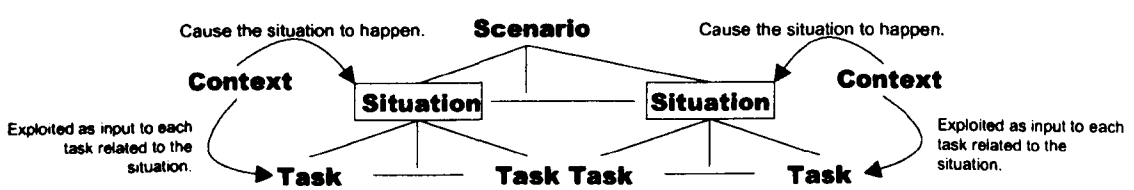


Figure B.1 Scenario, Situation, Task, and Context.

1. Scenario

1.1 Observation.

1.2 Situations.
Identify the context that causes the situation to happen.

2. User model (Lightweight)

2.1 Role in performing tasks.
Identify social context about the user in a situation. It can specify what tasks that the user can perform in her role in a situation.

2.2 Novice user in terms of the use of non-traditional sensors.

3. Task analysis

3.1 Non-computer supported user tasks.

3.2 Computer supported user-application tasks.
Convert the understanding of the user tasks into an appropriate human- computer interaction by transforming user tasks to computer application tasks. This would also decompose the abstract context obtained in step 1.2 into context that can be used as the input to each task.

4. Operating environment

4.1 Context acquisition.

4.2 Prototyping design concern
The adaptation of future sensor technology should be taken into account.

5. Action

Use context identified in the step 1.2, 3.1, and 3.2 to determine the situation and assign actions to respond the situation

We think things happened or is happing in the world is based on a certain context or a set of context. As mentioned in this chapter, context is words that come before and after a word, phrase, statement, etc, helping to show what its meaning is. We may consider an application is the word, phrase, or statement that is encompassed by other words, context, which help describe the meaning, function, of the application. An application is designed for a particular purpose that helps its user to achieve a certain goal in a specific situation. For instance, the goal of a game is to help the user have fun during the interaction with the game application. Designers hold the reasons that a particular situation happens and assign relevant actions in their designing application when the situation happens. Figure B.2 illustrate how a designer might consider a context-sensitive application. Ideally, users know what information they should provide when interacting with an application. The consensus is that users usually expect that an application can do appropriate tasks for them if they choose the correct application and provide information that the application requires explicitly. Some authors argued that each application can be thought of as establishing a context for user action (Lieberman and Selker, 2000). An application expresses what actions it can perform to its user and what it can operate upon. However, many applications are designed without a specific context in mind. For example, email was designed for a certain set of users but has been exploited by people doing very different ways. In the

real world, the situation of a person is always changing. An application cannot cover every situation that a user may encounter in her daily activity. What we would like to emphasis is that what an application can do is based on fixed context categories that can depict its user's current situation. Changing from one application to another means the situation is different from previous one passed and so is the underlying context. Our design process simply focus on how to design a context-sensitive mobile computing application based on a limited number of predefined scenario instead of designing an architecture that chains a bunch of applications together and dynamically selects and provides applications for the user based on her predicted information need in a situation (Michahelles, 2000, Schilit, 1995).

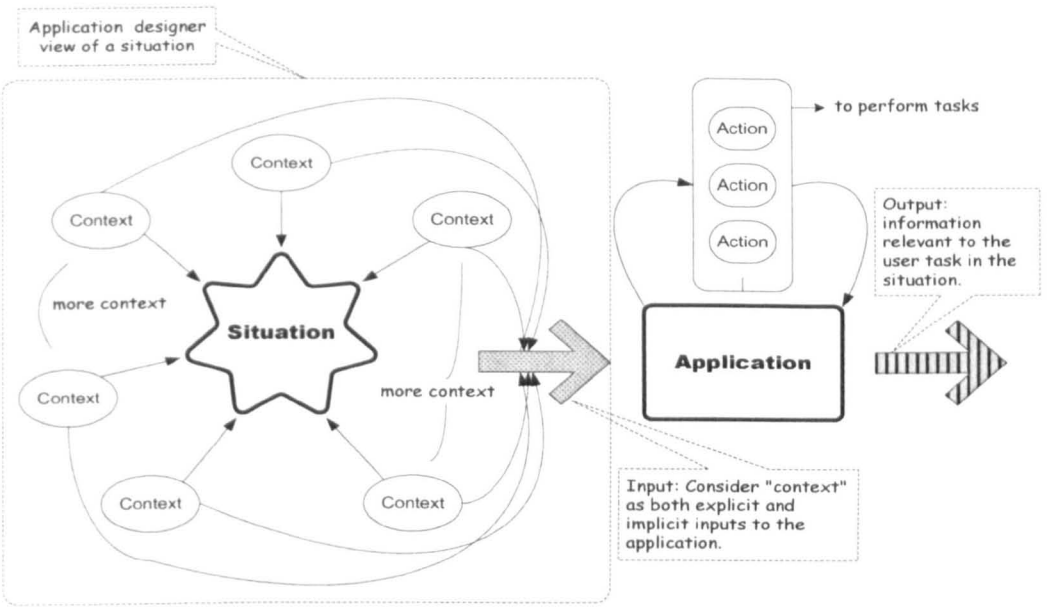


Figure B.2 Designer's view of a context-sensitive application

REFERENCES

Abowd, G., Battestini, A. and O'Connell, T. (2002). *The Location Service: A framework for handling multiple location sensing technologies*, College of Computing & GVU Centre, Georgia Institute of Technology, April.

Adams, D. (2002). *Programming Jabber,* 1st edition. O'Reilly, Sebastopol, CA, USA.

Allen, J. (2000). *Merging Mobility and Middleware, Build a lightweight, easy-to-implement architecture for accessing distributed objects from an Internet-enabled PDA*, http://www.devx.com/upload/free/features/javapro/2000/07jul00/ja0007/ja0007.asp.

Anhalt, J., Smailagic, A., Siewiorek, D. P., Gemperle, F., Salber, D., Weber, S. M., Beck, J. and Jennings, J. (2001). *Toward Context Aware Computing: Experiences and Lessons, IEEE Intelligent Systems and Their Applications,* **16,** pp. 38-46.

Annett, J. and Duncan, K. D. (1967). *Task analysis and training design, Psychological Review,* **41,** pp. 369-406.

Asthana, A., Cravatts, M. and Krzyzanowski, P. (1994). *An Indoor Wireless System for Personalized Shopping Assistance, IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, U.S., 1994.

Axelson, J. (1998). *Programming and Circuits for RS-232 and RS-485 Links and Networks,* Lakeview Research, 1st edition Madison, USA.

Bahl, P. and Padmanabhan, V. N. (2000). *Radar: An in-building RF-based user location and tracking system, IEEE Infocom 2000*, pp.775-784. Tel-Aviv, Israel, 2000.

Baldonado, M., Cousins, S., Gwizdka, J. and Pacpcke, A. (2000). *Notable: At the Interaction of Annotations and Handheld Technology*, Thomas, P. and Gellersen, H. W., *HUC 2000*, pp.143-156. Springer-Verlag, Bristol, U.K., 2000.

Bennett, J., Butler, K. A. and Whiteside, J. (1989). *Usability Engineering, CHI'89*, pp. 24-36 Austin, TX, 1989.

Bjork, S., Falk, J., Hansson, R. and Ljungstrand, P. (2001). *Pirates! - Using the Physical World as a Game Board., Interact 2001, IFIP TC.13 Conference on Human-Computer Interaction*, Tokyo, Japan, 2001.

Bloch, C. and Bodoff, S. (2001). *The Java Tutorial. Trail: Servlets,* http://java.sun.com/docs/books/tutorial/servlets/index.html

Boling, D. M. (2000). *Programming Microsoft Windows CE,* 1st edition, Microsoft Press International.

Broadbent, J. and Marti, P. (1997). *Location Aware Mobile Interactive Guides: usability issues, Fourth International Conference on Hypermedia and Interactivity in Museums (ICHIM97)*, pp.88-98. Paris, 1997.

Brown, P. J. (1998). *Triggering information by context, Personal Technologies, http://www.cs.ukc.ac.uk/pubs/1998/591.*

Brown, P. J. and Jones, G. J. F. (2002). *Exploiting contextual change in context-aware retrieval, ACM Symposium on Applied Computing (SAC2002)*, pp.650-656. Madrid, 2002.

Burrell, J. and Gay, G. (2001). *Collectively Defining Context in a Mobile, Networked Computing Environment, CHI 2001*, Seattle, 2001.

Burrell, J. and Gay, G. (2002). *E-graffiti: Evaluation Real -World Use Of a Context-Aware System, Special Issue on Universal Usability*, 2002

Buschmann (1996). *Pattern-oriented Software Architecture: a System of Patterns*, John Wiley & Sons.

Butler, K. A. (1996). *Usability engineering turns 10, Interactions*, **3**, pp. 58-75.

Byun, H. E. and Cheverst, K. (2001). *Exploiting User Model and Context-Awareness to Support Personal Daily Activities, Workshop on User Modelling for Context-Aware Applications*, pp.22-30. Sonthofen, Germany, 2001.

Cambridge, AT&T Laboratory (1992). *Active Badge on ``Beyond 2000''*, *http://www.uk.research.att.com/pub/videos/qsif-200/beyond-qsif-200.mpg*, 1992

Cambridge, AT&T Laboratory (1992). *The Active Badge System*, *http://www.uk.research.att.com/pub/videos/qsif-200/badge-qsif-200.mpg*, 1992

Cambridge, AT&T Laboratory (1994). *The Teleporting System*, *http://www.uk.research.att.com/pub/videos/qsif-200/teleport-qsif-200.mpg*, 1994

Cambridge, AT&T Laboratory (1998). *Ultrasonic Location Sensing*, *http://www.uk.research.att.com/pub/videos/qsif-200/bat-qsif-200.mpg*, 1998

Cambridge, AT&T Laboratory (1998). *Virtual Network Computing*, *http://www.uk.research.att.com/pub/videos/qsif-200/vnc-qsif-200.mpg*, 1998

Cambridge, AT&T Laboratory (2001). *Batportal*, *http://www.uk.research.att.com/ar/*, 2001

Cambridge International Dictionary of English Online Version (1999). http://dictionary.cambridge.org/

Carey, M. S., Stammers, R. B. and Astley, J. A. (1989). In *Task Analysis for Human-Computer Interaction* (Ed, Diaper, D.) Ellis Horwood, pp. 56-74.

Carroll, J. M. (2000). *Five reasons for scenario-based design, Interacting with Computers*, **13**, pp. 43-60.

Carroll, J. M. (2000). *Making Use Scenario-Based Design of Human-Computer Interactions*, The MIT Press, Cambridge, Massachusetts.

Carroll, J. M. and Rosson, M. B. (1999). *Getting Around the Task-Artifact Cycle: How to Make Claims and Design by Scenario, ACM Transactions on Information Systems (TOIS)*, **10**, pp. 181-212.

Castro, P., Ajmera, D., Castro, P., Kremenek, T. and Mani, M. (2001). *The Nibble Location System, http://mmsl.cs.ucla.edu/nibble/21.05.2001*

Caswell, D. and Debaty, P. (2000). *Creating Web Representations for Places*, Thomas, P. and Gellersen, H. W., *HUC2K*, pp.114-126. Springer-Verlag, Bristol, U.K., 2000.

Chan, W. (2000). *DealFinder: A Collaborative, Location-Aware Mobile Shopping Application*, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology

Chan, W. (2001). *Project Voyager: Building an Internet Presence for People, Places, and Things*, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology

Chen, G. and Kotz, D. (2001). *A Survey of Context-Aware Mobile Computing Research*, Dept. of Computer Science, Dartmouth College

Chen, G. and Kotz, D. (2001). *SOLAR: Towards a Flexible and Scalable Data-Fusion Infrastructure for Ubiquitous Computing, UbiTools workshop at UbiComp 2001*, Atlanta, US, 2001.

Chen, G. and Kotz, D. (2002). *Solar: An Open Platform for Context-Aware Mobile Applications, First International Conference on Pervasive Computing (Pervasive 2002)*, pp.41-47. Zurich, Switzerland, 2002.

Cheng, Y-M. and Johnson, C. (2001). *The Reality Gap: Pragmatic Boundaries of Context Awareness*, Blandford, A., Vanderdonckt, J. and Gray, P., *IHM-HCI 2001*, pp.412-427. Springer, Lille, 2001.

Cheverst, K. (1999). *Development of a Group Service to Support Collaborative Mobile Groupware*, PhD Thesis, 1999, Computing Science, Lancaster University, Lancaster

Cheverst, K., Davies, N., Mitchell, K., Friday, A. and Efstratiou, C. (2000). *Developing a context-aware electronic tourist guide: some issues and experiences, CHI'00*, pp.17-24. ACM Press, Netherlands, 2000.

Cheverst, K., Mitchell, K. and Davies, N. (2002). *Exploring Context-aware Information Push, Personal and Ubiquitous Computing*, **6**, 276-281.

Coschurba, P., Baumann, J., Kubach, U. and Leonhardi, A. (2001). *Metaphors and Context-Aware Information Access, Personal and Ubiquitous Computing*, **5**, 16-19.

Crease, M., Gray, P. and Cargill, J. (2001). *Using location information in an undergraduate computing science laboratory support system, Location Modelling for Ubiquitous Computing, Ubicomp 2001*, Atlanta, U.S., 2001.

Dana., P. H. (2000). *Global positioning system overview, http://www.colorado.edu/geography/gcraft/notes/gps/gps.html*

Davies, N., Cheverst, K., Mitchell, K. and Efrat, A. (2001). *Using and Determining Location in a Context-Sensitive Tour Guide, IEEE Computer*, **34**, 35-41.

Davies, N., Cheverst, K., Mitchell, K. and Friday, A. (1999). *'Cache in the Air': Disseminating Tourist Information in the Guide System, 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '99)*, New Orleans, Louisiana, U.S., 1999.

Dennis, B. (2001). *On Integrating First Person Shooter Games and Ubiquitous Environments, UbiComp2001*, Atlanta, U.S., 2001.

DeVaul, R. W. and Dunn, S. (2001). *The Context Aware Cell Phone Project, http://www.media.mit.edu/wearables/mithril/phone.html*

Dey, A. (2000). *Providing Architectural Support for Building Context-Aware Applications*, PhD Thesis, 2000, Computer Science, Georgia Institute of Technology, Atlanta

Dey, A. K. (2001). *Understanding and Using Context, Personal and Ubiquitous Computing*, **5**, pp. 4-11.

Dix, A., Finlay, J., Abowd, G. and Beale, R. (1998). *Human-Computer Interaction*, Prentice Hall Europe, 2nd edition.

Dix, A., Rodden, T., Davies, N., Trevor, J., Friday, A. and Palfreyman, K. (2000). *Exploiting space and location as a design framework for interactive mobile systems, ACM Transactions on Computer-Human Interaction*, **7**, pp. 285-321.

Dourish, P. (2001). *Seeking a Foundation for Context-Aware Computing, Human-Computer Interaction*, **16**, 1-15.

Dryer, D. C., Eisbach, C. and Ark, W. S. (1999). *At what cost pervasive? A social computing view of mobile computing systems, IBM SYSTEM JOURNAL,* **38,** pp. 652-676.

Duncan, K. D. (1972). In *Strategies for programmed instruction: an educational technology*(Ed, Hartly, J.) Butterworths, London, pp. 19-81.

E., P. and Bhargava, B. (1994). *Building Information Systems for Mobile Environments, 3rd International Conference on Information and Knowledge Management,* pp.371-378. Washington, D.C., 1994.

Espinoza, F., Persson, P., Fagerberg, P., Sandin, A. and Coster, R. (2001). *GeoNotes: Social and Navigational Aspects of Location-Based Information Systems,* G. Abowd, B., Shafer, *Ubicomp2001,* pp.2-17. Springer, Atlanta, Georgia, 2001.

Falk, J. (2001). *The Nexus: Computer Game Interfaces for the Real World, UbiComp2001,* Atlanta, U.S., 2001.

Faulkner, C. (1998). *The Essence of Human-Computer Interaction,*1st edition, Prentice Hall Europe.

Forman, G. H. and Zahorjan, J. (1994). *The Challenges of Mobile Computing, IEEE Computer,* **27,** 38-47.

Galani, A. and Chalmers, M. (2002). *Can You See Me? Exploring Co-Visiting Between Physical and Virtual Visitors, Museums and the Web 2002, Archives & Museum Informatics,* Boston, U.S., 2002.

Gellersen, H.-W. and Gaedke, M. (1999). *Object-Oriented Web Application Development, IEEE Internet Computing,* pp. 60-68.

Gibson, E. (2002). *Exploring off-the-shelf Infrared Technology to provide Location Indices in a Retail Environment,* Department of Computing Science, University of Glasgow, April

Greenhalgh, C., Benford, S., Rodden, T., Anastasi, R., Taylor, I., Flintham, M., Izadi, S., Chandler, P., Koleva, B. and Schnadelbach, H. (2002). *Augmenting Reality Through Coordinated Use of Diverse Interfaces,* Equator - Nottingham, November

Gregory Abowd, C. G. A., Jason Hong, Sue Long, Rob Kooper, Mike Pinkerton (1997). *Cyberguide: A Mobile Context-Aware Tour Guide, ACM Wireless Networks,* **3,** pp. 421-433.

Harold, E. R. (1999) *Java I/O,* O'Reilly, 1st edition, Sebastopol, CA, USA.

Harter, A. and Hopper, A. (1998). *A Distributed Location System for the Active Office,* AT&T Laboratories Cambridge, January

Harter, A., Hopper, A., Steggles, P., Ward, A. and Webster, P. (1999). *The Anatomy of a Context-Aware Application, ACM/IEEE MobiCom,* pp.59-68. Seattle, U.S., 1999.

Headon, R. and Curwen, R. (2001). *Ubiquitous Game Control, UBICOMP Workshop on Designing Ubiquitous Computing Games,* Atlanta, GA, USA, 2001.

Hightower, J. and Borriello, G. (2001). *Location Sensing Techniques,* Department of Computer Science and Engineering, University of Washington, July

Hightower, J. and Borriello, G. (2001). *A Survey and Taxonomy of Location Sensing Systems for Ubiquitous Computing,* Department of Computer Science and Engineering, University of Washington

Hix, D. and Hartson, H. R. (1993). *Developing User Interfaces: ensuring usability through product and process.,* J. Wiley and sons.

Hohl, F., Kubach, U., Leonhardi, A., Rothermel, K. and Schwehm, M. (1999). *Next Century Challenges: Nexus - An Open Global Infrastructure for Spatial-Aware Applications, Mobicom '99*, pp.249-255. ACM Press, Seattle, U.S., 1999.

Hong, J. I. and Landay, J. A. (2001). *An Infrastructure Approach to Context-Aware Computing, Human-Computer Interaction (HCI) Journal,* **16,** 23.

INC., I. S. (1997). *A Primer on Remote Control Technology*, Innotech Systems INC, Boston, USA.

IrDA (2001). *Infrared Data Association Serial Infrared Physical Layer Specification*, Infrared Data association, 30 May

Ishii, H. and Ullmer, B. (1997). *Tangible bits: towards seamless interfaces between people, bits and atoms*, Pemberton, S., *Human Factors and Computing Systems*, pp.234-241. ACM Press, Atlanta, U.S., 1997.

Jameson, A. (2001). *Modelling both the Context and the User, Personal and Ubiquitous Computing,* **5,** 29-33.

Java (2002). *Java Tutorial Online Version, http://java.sun.com/tutorial*

Johnson, C. and Cheng, Y-M. (2002). In *HUMAN FACTORS AND WEB DEVELOPMENT*, 2nd edition, eds. Ratner, J. Lawrence Erlbaum, New York, pp. 241-264.

Kieras, D. E. and Polson, P. G. (1985). *An approach to the formal analysis of user complexity, International Journal of Man-Machine Studies,* **22,** pp. *365-394.*

Kindberg, T. and Barton, J. (2001). *A Web-based nomadic computing system, Computer Networks (Amsterdam, Netherlands),* **35,** pp. 443-456.

Kindberg, T., Barton, J., Morgan, J., Becker, G., Caswell, D., Debary, P., Gopal, G., Frid, M., Kirshnan, V., Morris, H., Schettino, J., Serra, B. and Spasojevie, M., (2000). *People, Place, Things: Web Presence for the Real World*, Hewlett-Packard Laboratories

Korkea-aho, M. (2000). *Context-Aware Applications Survey*, Department of Computer Science, Helsinki University of Technology, *http://www.hut.fi/~mkorkeaa/doc/context-aware.html*

Kortuem, G., Bauer, M., Heiber, T. and Segall, Z. (1999). *Netman: The Design of a Collaborative Wearable Computer System, Journal on Mobile Networks and Applications,* **4.** pp. 245-256.

Leonhardi, A. and Bauer, M. (2000). *The VIT-System: Experiences with Developing a Location-Aware System for the Internet, Workshop on Infrastructure for Smart Devices - How to Make Ubiquity an Actuality,HUC 2K*, Bristol, U.K., 2000.

Lieberman, H. and Selker, T. (2000). *Out of Context: Computer systems that adapt to, and learn from, context, IBM SYSTEM JOURNAL,* **39,** pp.617-632.

Lijungstrand, P. (2001). *Context Awareness and Mobile Phones, Personal and Ubiquitous Computing,* **5,** 58-61.

Long, S., Kooper, R., Abowd, G. D. and Atkeson, C. G. (1996). *Rapid prototyping of mobile contextaware applications: The cyberguide case study, conference on Human Factors in Computing Systems ,CHI'96*, pp.97-107. 1996.

Lueg, C. (2001). *ON CONTEXT-AWARE ARTIFACTS AND SOCIALLY RESPONSIBLE DESIGN, Computer-Human Interaction (OzCHI 2001)*, pp.1-6. Fremantle (Perth), Australia, 2001.

MacColl, I., Brown, B., Benford, S. and Chalmers, M. (2002). *Shared Visiting in EQUATOR City, Collaborative Virtual Environments 2002*, Bunn, 2002.

MacIntyre, B. (2001). *Context-Aware Personal Augmented Reality*, Beaudouin-Lafon, M. and Mackay, W. E., *Workshop at CHI 2000, ACM Conference on Human Factors in Computing Systems*, Aarhus N - Denmark, 2001.

MacKenzie, I. S. and Zhang, S. Z. (1999). *The Design and Evaluation of a High-Performance Soft Keyboard, ACM Conference on Human Factors in Computing Systems - CHI '99*, pp.25-31. ACM, Nwe York, 1999.

Manser, M. H. (2000). *Oxford Advanced Learner's English-Chinese Dictionary*, 4ed.

Marmasse, N. and Schmandt, C. (2000). *Location-Aware Information Delivery with ComMotion*, Thomas, P. and Gellersen, H. W., *HUC2K*, pp.157-171. Springer-Verlag, Bristol, U.K., 2000.

Masui, T. and Siio, I. (2001). *Real-World Graphical User Interfaces*, Thomas, P. and Gellersen, H. W., *HUC 2000*, pp.72-84. Springer-Verlag, Bristol, U.K., 2001.

Michahelles, F. (2000). *Designing an Architecture for Context-Aware Service Selection and Execution*, 2000, Institut Fur Informatik, Lehr- und Forschungseinheit fur Kommunikationssysteme und Systemprogrammierung, Munchen

Mohamed, A. (1998). *SC546 Project, An analytical study ofIR signals used by a SONY remote control, http://cgl.bu.edu/GC/shammi/ir/.*

Myrhaug, H. I. (2001). *Towards Life-long and Personal Context Spaces*, Gross, T. and Specht, M., *Workshop on User Modelling for Context-Aware Applications*, 2001.

Navarre, D., Palanque, P., Paterno, F., Santoro, C. and Bastide, R. (2001). In *Interactive Systems Design, Specification, and Verification*(Ed, Johnson, C.) Springer, New York, pp. 88-113.

network, c. d. s. (2002). *Web Presence Manager, http://cooltown.hp.com/dev/reference/coolbase/ds-wpm.asp,18/.*

Nievergelt, J. and Hinrichs, K. H. (1993). *Algorithms & Data Structure*, 1st edition, Prentice-Hall, Inc., New Jersey.

Oppermann, R. and Specht, M. (1999). *A nomadic Information System for Adaptive Exhibition Guidance, ICHIM99, International Cultural Heritage Meeting*, pp.103-109. Washington, D.C., U.S., 1999.

Oppermann, R. and Specht, M. (2000). *A Context-Sensitive Nomadic Exhibition Guide*, Thomas, P. and Gellersen, H. W., *HUC2K*, pp.127-142. Spring-Verlag, Bristol, U.K., 2000.

Palanque, P., Bastide, R. and Senges, V. (1995). In *Engineering for Human-Computer Interaction*(Eds, Bass, L. J. and Unger, C.) Chapman and Hall, pp. 189-212.

Pascoe, J. (1997). *The Stick-e Note Architecture: Extending the Interface beyond the User, Intelligent User Interfaces (IUI'97)*, pp.261-264. ACM Press, Orlando, Florida, United States, 1997.

Pascoe, J., Ryan, N. and Morse, D. (1998). *Human-Computer-Giraffe Interaction: HCI in the Field*, Johnson, C., *First Workshop on HCI for Mobile Devices*, pp.48-57. Glasgow, U.K., 1998.

Pering, T. P., C. (2001). *Mercantile: Social Interaction Using a Mobile Computing Platform, UbiComp2001*, Atlanta, U.S., 2001.

Pooley, R. and Stevens, P. (1999). *Using UML, Software Engineering with Objects and Components*, Addison-Wesley.

Punnoose, R. J., Tseng, R. S. and Stancil, D. D. (2001). *Experimental Results for Interference between Bluetooth and IEEE 802.11b DSSS Systems, IEEE Vehicular Society Conference*, pp.55-67. Tel Aviv, Israel, 2001.

Raab, F., Blood, E., Steiner, T. and Jones, H. (1979). *Magnetic Position and Orienting Tracking System, IEEE Transactions on Aerospace and Electronic Systems,* **AES-15,** 709-718.

Microsoft Research (2000). *RADAR in Action, http://research.microsoft.com/~bahl/MS_Projects/Videos/Mobicom.mpg*

Sage, M. (2001). *Declarative Support for Prototyping Interactive Systems*, PhD Thesis, 2001, Department of Computing Science, University of Glasgow, Glasgow, U.K.

Salber, D., Dey, A., Orr, R. and Abowd, G. (2000). *Designing for Ubiquitous Computing: A Case Study on Context Sensing*, Graphics, Visualization, and Usability Center, Georgia Institute of Technology.

Schigeoka, I. (2002). *Instant Messaging in Java,* 1st edition, Manning Publications Co.

Schilit, W. N. (1995). *A System Architecture for Context-Aware Mobile Computing*, PhD Thesis, 1995, Computer Science, Columbia University, New York

Schmandt, C., Marmasse, N., Marti, S., Sawhney, N. and Wheeler, S. (2000). *Everywhere messaging, IBM SYSTEM JOURNAL,* **39,** 660-677.

Schmidt, A. (2000). *Implicit Human Computer Interaction Through Context, Personal Technologies,* **4,** pp. 191-199.

Schmidt, A. and Beigl, M. (1998). *New Challenges of Ubiquitous Computing and Augmented Reality, Cabernet Radicals Workshop 98*, Porto, 1998.

Schmidt, A., Beigl, M. and Gellersen, H.-W. (1998). *There is more to context than location, Workshop on Interactive Applications of Mobile Computing (IMC98)*, Rostock, Germany, 1998.

Schmidt, A., Takaluoma, A. and Mntyjrvi, J. (2000). *Context-Aware Telephony over WAP*, pp.225-229. Springer-Verlag, London, Ltd., 2000.

Schneider, J. and Kortuem, G. (2001). *How to Host a Pervasive Game - Supporting Face-to-Face Interactions in Live-Action Roleplaying, UbiComp 2001*, Atlanta, GA, USA, 2001.

Scholtz, J. (2001). *Evaluation Methodologies for Context-aware Computing, CHI 2001 Workshop on Distributed and Disappearing User Interfaces in Ubiquitous Computing*, Seattle, 2001.

Scott, B. (2002). *Using Portable Digital Assistants (PDAs) to support ubiquitous computing,* Department of Computing Science, University of Glasgow, April

Selker, H. Y. a. T. (2000). *Context-aware office assistant, 2000 International Conference on Intelligent User Interfaces*, pp.276-279. ACM Press, New Orleans, LA, U.S., 2000.

Selker, T. and Burleson, W. (2000). *Context-aware design and interaction in computer systems, IBM SYSTEM JOURNAL,* **39,** pp. 880-891.

Shepherd, A. (1989). In *Task Analysis for Human-Computer Interaction*(Ed, Daiper, D.) Ellis Horwood, pp. 15-55.

Spasojevic, M. and Kindberg, T. (2001). *A Study of an Augmented Museum Experience*, HP Laboratories, 19.07.2001

Spohrer, J. C. (1999). *Information in Places, Pervasive Computing,* **38,** pp. 602-628.

Stein, L. and Stewart, J. (2001). *W3C, The World Wide Web Security FAQ*, W3C, 16 October

Suvak, D. W., Megowan, P. J., Young, L., Kondo, T., Esashi, M., Matsumoto, M., SAITO, R. and Nykanen, P. (1995). *IrCOMM: Serial and Parallel Port Emulation over IR (Wire Replacement)*, IrDA, 7 November

Agilent Technology (2001). *Agilent IrDA Data Link Design Guide*, Agilent Technology, 5988-1772EN, 25 May

Tuulari, E. (2000). *Context aware hand-held devices*, MSc Thesis, VTT ELECTRONICS, NETWORKING RESEARCH, TECHNICAL RESEARCH CENTRE OF FINLAND, OULU

Venners, B. (1999). *Inside the JAVA 2 Virtual Machine*, The McGraw-Hill Companies, Inc., 2ed.

Wabasoft (2001), *http://www.wabasoft.com*.

Want, R., Fishkin, K. P., Gujar, A. and Harrison, B. L. (1999). *Bridging Physical and Virtual Worlds with Electronic Tags*, *ACM CHI '99*, pp.370-377. Pittsburgh, PA, U.S., 1999.

Want, R., Hopper, A., Falcao, V. and Gibbons, J. (1992). *The Active Badge Location System*, *ACM Transactions on Information Systems*, **10**, pp. 91-102.

Want, R. and Russell, D. M. (2000). *Ubiquitous Electronic Tagging*, *IEEE Distributed Systems Online*, **1**. pp. 15-21

Want, R., Schilit, B. N., Adams, N. I., Gold, R., Petersen, K., Goldberg, D., Ellis, J. R. and Weiser, M. (1995). *The ParcTab Ubiquitous Computing Experiment*, Xerox Parc, *http://citeseer.nj.nec.com/535.html*

Ward, A., Jones, A. and Hopper, A. (1997). *A New Location Technique for the Active Office*, *IEEE Personnel Communications*, **4**, pp. 42-47.

Weiser, M. (1993). *Some computer science issues in ubiquitous computing*, *Communications of the ACM*, **36**, pp. 75-84.

Weiser, M. (1996). *Ubiquitous Computing*, *http://www.ubiq.com/hypertext/weiser/ubihome.html*, *17.03.1996*.

Weiser, M. (1998). *The Invisible Interface: Increasing the Power of the Environment through Calm Technology*, *Cooperative Buildings Integrating Information, Organization, and Architecture (CoBuild'98)*, Darmstadt, Germany, 1998.

Weiser, M. and Brown, J. S. (1996). *The Coming Age of Calm Technology*, Xerox PARC, 5 October

Youll, J., Morris, J., Krikorian, R. and Maes, P. (2000). *Impulse: Location-based Agent Assistance*, *Fourth International Conference on Autonomous Agents*, Barcelona, Spain, 2000.

Zimmerman, T. G. (1999). *Wireless networked digital devices: A new paradigm for computing and communication*, *IBM SYSTEM JOURNAL*, **38**, pp. 566-574.