Abstract

Supporting The Development of Mobile Context-Aware Systems

Keith Mitchell

Computing Department,
Lancaster University,
England, U.K.

Submitted for the degree of Doctor of Philosophy,
January, 2002

The recent convergence of mobile and context-aware systems has seen a considerable rise in interest in applications that exploit aspects of the operating environment to offer services, tailor application behaviour or trigger adaptation. However, as a result of the lack of generic mechanisms for supporting user mobility and context awareness within dynamic environments, context-aware applications remain very difficult to build and developers must deal with a wide range of issues which may be incidental to the development of new applications. As a result of these issues, few mobile context-

aware applications exist outside the boundaries of the research laboratory and even fewer have been realistically deployed and evaluated in *real* world settings.

In addition, traditional context-aware applications are poorly suited to highly mobile or distributed environments and often unable to tolerate a rapidly changing execution environment, or take advantage of the availability of new services. Moreover, existing approaches to the development of context-aware applications are, in general, highly reliant upon the underlying infrastructure. Consequently application developers must build their applications with specific environments (indoor or outdoor) or services in mind. As a consequence, this makes applications less flexible, that is, portable across different end-systems and operating environments, difficult to extend or evolve their functionality, and crucially, unable to tolerate use in a fluctuating service environment.

This thesis describes the design and implementation of a *context-service* based architecture to support the development of mobile context-aware applications designed for use in distributed environments, addressing the salient challenges that this involves. The approach is validated using a real world mobile context-aware application, the Lancaster GUIDE system. The GUIDE system is used as a vehicle for research into the area of mobile context-awareness and, through a retrospective analysis and evaluation of the GUIDE approach, the ideas presented in this thesis have been established. This thesis demonstrates how a re-engineered version of the GUIDE prototype benefits from the service based approach in terms of its flexibility. More specifically, the re-engineered version of GUIDE is better able to operate in a rapidly changing execution environment. The overall results provide a valuable insight into the effectiveness and applicability of the service based approach to the mobile domain in general, and suggests that context-services provide a useful model for the development and experimentation of a wide range of context-aware systems.

Declaration

The overall concept of the context-service based architecture to support context-aware applications within dynamic mobile environments was my own. The use of the GUIDE and GUIDE II EPSRC funded research projects (grant numbers GR/L05280 and GR/CSA7622 respectively) provided an invaluable and unique research environment with which to experiment and develop the ideas presented in this thesis. The design and implementation of the context service provider (CSP) and prototype context-services was conducted by myself. The concept of the context service was established and developed during a number of extensions to the original GUIDE implementation and benefited from discussions with colleagues from Lancaster involved with the GUIDE projects. However, the GUIDE and GUIDE II prototype implementations detailed in chapters three and six respectively were implemented by myself.

The work reported in this thesis has not been previously submitted for a degree, in this or any other form.

Acknowledgements

The process of completing a Ph.D. thesis is a very personal journey, however it does require the help and support of many others to successfully complete the task. Here I wish to express my thanks to all who have been involved in supporting me during the completion of my Ph.D.

I would like to begin by thanking Professor Nigel Davies for providing me with the initial opportunity to work within such a dynamic and exciting team of researchers and, furthermore, for all the help, support, motivation and opportunities he has provided me with over the years. Secondly, I would like to thank Professor Gordon Blair for agreeing to continue my supervision during Nigel's absence and for his continued advice, encouragement, commitment and constructive criticism throughout the preparation of this document. I would also like to extend my thanks to my closest work colleague (and pseudo-supervisor!) Dr. Keith Cheverst, whose informal support and words of encouragement have been very significant to my progress both during the implementation and writing up phases of my research. The numerous coffees, B.L.T.s and games of pool over the last few years have certainly helped! My other colleagues within the Distributed Multimedia Research Group (DMRG) at Lancaster must also be thanked for providing a constructive environment in which the ideas presented within this thesis have been established. In particular, Dr Adrian Friday for his support during the early stages of the GUIDE project and Matthew Storey, with whom I have not only shared office space with but the whole Ph.D. process. Finally, since the GUIDE project was a collaborative venture between Lancaster University

and Lancaster City Council, I would like to thank all involved in the project, particularly, the industrial partners, HP Labs (Bristol) and Lucent Technologies.

I fully appreciate the support of my parents, Lisa and Allan, in providing me with a good educational foundation and, perhaps most importantly, the opportunity to attend University. The desire for learning gained as a child has led me to academia and provided me with a strong basis for establishing my future research career. Little did I realise back then as a child how useful my early exposure to computer technology and my early programming skills would prove to be!

Many sacrifices and compromises have to be made during the completion of a Ph.D. and I would like to express my deepest gratitude to my partner Fiona who has, more than *anyone*, had to make sacrifices to enable me to complete my Ph.D., often at the cost of her own Ph.D. progress. Furthermore, her consistent love, support and friendship has made the process of writing this thesis tolerable. During the many times I have doubted myself and my work, she has provided the encouragement and motivation to push on. I simply could not have reached this point without her and for that I am eternally grateful. As a reward, both you and my two lovely daughters Isla and Leila have had to bear the brunt of all the ups and downs that are associated with completing a Ph.D. (sorry!). I only hope that I can offer you (Fiona) a similar level of support throughout your Ph.D. research and thesis write-up.

To you *all, thanks!*

How do (will) people use mobile systems and applications? [Satyanarayanan, 96].

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Overview

Mobile computing currently plays an increasingly significant role in everyday life due to advances in personal computing and communication technologies. Moreover, the availability of low cost sensing technologies allows applications to monitor and use situational information, or *context*, gathered from their operating environments, for example, to aid a tourist navigating a city using location information captured from a GPS [GPS,01] device. Applications that take into consideration environmental factors such as location, identity and time are termed *context-aware* applications.

Existing context-aware applications are, in general, poorly suited to mobile or distributed environments and often unable to tolerate a rapidly changing execution environment, or take advantage of the availability of new services. Moreover, existing approaches to developing context-aware applications are typically dependent on the underlying infrastructure and thus applications are built with particular environments (indoor or outdoor), application domains and infrastructures (computational services or sensors) in mind [Dey,00c]. Consequently, this makes applications less flexible, and difficult to extend or evolve their functionality should the existence of new entities be detected. For example, an application utilising GPS

for location awareness may have to be re-written in order to utilise an alternative technology such as Active Badges [AT&T,00], [Want,92].

This thesis explores the pertinent issues relating to the development of context-aware applications designed for use in dynamic environments and proposes to address the salient challenges detailed above by adopting a *context-service* based approach. This approach affords a layer of abstraction between applications, system components and underlying technologies. Novel to this approach is that an application is able to discover dynamically and select context services based on arbitrary contextual information (e.g. user stipulated context constraints). Furthermore, decomposing applications into independent services and embedding them in to the infrastructure provides a foundation for other applications. As a result, applications may be smaller in size, since core functionality resides within the infrastructure and can be shared, instead of being single monolithic and self-contained entities. The decomposition of applications into independent services is a classical solution to address the above issues in distributed systems [Blair,97] and this thesis demonstrates how a number of prototype context-aware applications benefit from the service based approach in terms of flexibility. More specifically, the use of context services provides applications with the necessary resources dynamically but are independent from any particular implementation and, as a direct result, can be independently developed.

The remainder of this chapter provides an overview of the enabling technologies for the field of mobile computing before describing in detail some of the relevant terms used above. This chapter concludes by refining the aims and scope of the research and summarises the contributions of this thesis.

1.2 Underlying Technologies

Developments in both the affordability and complexity of mobile computing devices and wireless communications technologies within the consumer marketplace engenders a higher degree of "*anytime-anywhere*" or *ubiquitous* access to distributed computing systems, information repositories and remote services. This section provides an outline of the enabling technologies for the contemporary mobile user. Since the work described in this thesis is technology independent and assumes an underlying infrastructure of highly portable multimedia devices, heterogeneous

communications and sensing technologies, an in-depth study of enabling technologies is not provided. More comprehensive surveys of the technologies described in this section can be found elsewhere in [Cheverst,99b], [Finney,99] and [Jose,01a].

1.2.1 Personal Devices

Within the portable computing market there now exists a spectrum of devices ranging from relatively large notebook computers, offering similar capabilities to that of their desktop counterparts, to small and discreet wearable computers, with more limited capabilities. The range of portable devices currently available include:

- **Notebooks and Sub-notebooks** Notebooks offer similar functionality to their desktop equivalents, with many high-end ranges aimed as desktop replacements. Sub-notebooks [Sony,01] offer increased portability with negligible reduction in system performance. However, the smaller form factor results in a reduction in screen real estate and expandability.

- **Digital Tablets** Digital tablets utilise a touch-sensitive screen and stylus for user input [Fujitsu,01]. They offer notebook performance, with increased screen real estate due to the omission of a keyboard, and can exploit standard desktop operating systems and development environments.

- **Handheld and Palm PCs** Handheld PCs (H/PCs) are small and lightweight portable devices available in a variety of form factors, each with different displays, peripherals and operating systems. Communication is often facilitated through IrDA [IrDA,98], and more recently, GSM [ECM,01], [Scourias,00] wireless LAN and Bluetooth cards [Bluetooth,99b].

- **Personal Digital Assistants (PDAs)** PDAs were originally designed to be very small, lightweight devices with simple applications such as calendar, notebook, address book and limited expandability. However, the current generation of PDAs, such as the Palm VII [Palm,00] or the Compaq iPAQ [Compaq,01], offer a variety of applications and support IrDA, GSM, WaveLAN and Bluetooth connectivity. Example applications include pocket versions of web browsers, email clients and multimedia players.

- **Smart Phones** A smart phone is a digital cellular phone that provides additional services including email, Internet access and personal information management (PIM), for example, the Nokia Communicator 9210 [Nokia,01]. The imminent deployment of 2.5 G and 3G broadband wireless networks will add video playback capabilities to these devices.

- **Wearable Computing** Wearable computing explores the potential of computer devices that are as unconsciously portable and personal as clothes or jewellery. Wearable devices afford more natural forms of interfaces, facilitating a richer variety of communications capabilities between humans and computers. The goal is to support common forms of human expression and leverage upon our implicit actions in the world, for example [Randell,00].

1.2.2 Communications

Wireless communications technologies available today can be categorised in the following ways:

- **Personal Area Network (PAN)** A personal-area network, in relation to computer networking, is the term used to represent the wireless interconnection of personal electronic devices [Braley,00] such as notebooks, handhelds, mobile phones using technologies such as Bluetooth [Haartsen,98].

- **Local Area Network (LAN)** A computer network that spans a relatively small geographic area, for example, a single room, a building or group of buildings.

- **Metropolitan Area Network (MAN)** Metropolitan-area networks are usually characterised by very high-speed connections using fibre optical cable or other digital media and are larger (geographically) than local-area networks, for example, connecting computers across a campus or city area.

- **Wide Area Network (WAN)** A computer network that spans a relatively large geographical area. Computers connected to a wide area network are often connected through public networks, such as the telephone system, although they can also be connected through leased lines or satellite systems.

The range of available Local Area Networking technologies can be grouped into two distinct classifications: infra-red (IR) and radio frequency (RF). Infra-red has become ubiquitous within the home audio/video (AV) market since the 1980's. Most IrDA implementations today are point-to-point (Direct IR), allowing bit rates of up to 4 Mbps over distances of approximately one metre. However, higher bit rate systems (10 Mbps) providing omni-directional (Diffuse IR) coverage also exist, for example, SpectrixLite [Spectrix,99]. IR systems are generally best suited to indoor environments since the technology cannot easily penetrate objects such as walls or even glass windows and is highly susceptible to ambient noise. Moreover, the point-to-point nature of the technology makes it difficult to communicate with multiple devices simultaneously and therefore establish ad hoc networks.

To address the problems of using IR in outdoor environments several RF technologies have been developed, since these are more able to penetrate objects like office partitions and walls. Today, a variety of commercial wireless LAN technologies are available including Lucent Orinocco [Orinocco,01], AiroNet [Cisco,01], Netwave [Xircom,01], Breezenet [Alvarion,01] and Nokia C111 [Nokia,00]. The bit rates offered by these technologies range from 2 Mbps to more than 11 Mbps up to a range of approximately 450 metres (omni-directional). These products offer either ISA, PCMCIA (type II) or USB interfaces and are easily installed into any modern portable end system. Wireless LANs, are designed primarily for enabling high bandwidth reliable communications within domestic, office [Bahl,00] or outdoor environments [Davies,98a], [Baker,96].

The latest RF technology to reach the consumer market is Bluetooth. This open specification with over 2000 consortium members including companies such as IBM, Motorola, Nokia, Intel and Microsoft has been designed to enable ad hoc wireless networking amongst portable computing and telecommunications devices, for example, PDAs, phones and notebooks. The current version specifies a data rate of 1 Mbps for ranges of up to 10 metres. A number of products are currently available from manufacturers such as Ericsson [Ericsson,01], Nokia [Nokia,99], Digianswer [Digianswer,00] and Palm [Palm,01]. The long term aim of the consortium is to have Bluetooth transceivers costing under 1 USD and for them to become ubiquitous amongst a wide range of electronic devices.

1.3 Sensing Technologies

The existence of low cost sensing technologies that enable a system to monitor aspects of its operating environment or its user, or more broadly, its *situation* provides a basis for developing mobile applications that can adapt to changes in context.

1.3.1 Location

Between 1989 and 1993 the United States Department of Defense (DoD) deployed the Global Positioning System (GPS), a satellite based navigation system funded and controlled by the U.S DoD. GPS uses a constellation of satellites and one-way communication as a means to determine user location based on a scheme using relative distances [GPS,01]. This system provides any user equipped with a GPS receiver with highly accurate positional information anywhere on earth. GPS receivers vary in form factor from wristwatches [Casio,01] to handheld receivers [Garmin,01] as well as solutions that allow connectivity with a PDA.

During a similar period (1989 to 1992) the Active Badge [Want,92] was developed at Olivetti. The active badge periodically emits a unique code (beacon) and these are received by a network of sensors placed around the building. A master station, polls the sensors for badge 'sightings', processes the data received from the sensors and presents the information visually, enabling relatively accurate positional information to be determined within an indoor environment. More recently, the ultrasonic location system, or Active Bat [Harter,99] developed by Cambridge University Computer Laboratory has made possible finer grained location within three dimensions (3D). In the Bat system an ultrasound beacon is emitted from a transmitter (known as a *bat*) which may be worn by a user or attached to objects (such as printers, chairs or tables). The time the beacon takes to reach a network of ceiling based receivers enables the distance between the bat and each receiver to be calculated. An accurate 3D position of the bat can be determined if three of more receivers are used and furthermore; if, for example, two or more bats are attached to an object, such as an office chair, then a bat's orientation can also be calculated.

1.3.2 Physical Environment

A range of relatively cheap devices capable of monitoring changes in the temperature and humidity of the environment are available from companies such as Dallas Semiconductors Corporation. For example, 1-Wire Weather Station [iButton,00b] can be used to monitor basic facets of the environment. Point Six Incorporated [Pointsix,01] and Kenelec [Kenelec,01] also offer products that are able to monitor environmental facets including wind direction, temperature, humidity, barometric pressure and rain level. Whilst individually these instruments may not provide a fully comprehensive representation of the environment, the combination of these attributes may be used to help applications adapt. For example, consider a user of an electronic tour guide with access to some environmentally sensed information. The tourist application may use information such as the time of day, the weather forecast and current light level to determine the likelihood of there being a spectacular sunset that day before directing the user to the best vantage point to view the sunset.

1.3.3 Physiological Environment

Monitoring the human body can also provide valuable information relating to the current state of the operating environment. Considering examples such as airline flight crew, train drivers or personnel operating heavy machinery, safe operation of these systems are dependant upon their personnel and therefore, may benefit from monitoring physiological factors that may be detrimental to their safe operation [Allanson,00]. Computing systems that relate to, arise from, or deliberately influence emotions or physiological facets are termed *affective computing* systems. MIT Media Lab are currently engaged in a number of research projects [Affective,00] that study information relating to the user's physical state or behaviour collected by a series of sensors worn on the human body. Using these sensors researchers have been able to gather information in a continuous way without interrupting users' normal work patterns. Example applications include being able to detect driver stress [Healey,00], the expression glasses [Scheirer,00], which allow a viewer to visualise expressions of confusion made by the wearer of the glasses, and AffQuake [Affective,00], an attempt to alter game play of ID Software's Quake II by monitoring signals from sensors attached to the player's physical body.

1.4 Introduction To Context

The preceding sections provided an overview of the underlying technologies for mobile computing and technologies that enable facets of the environment to be sensed or monitored. This information or *context* enables a system to have *awareness* of its *situation*. Thus, *context-aware* applications can be defined as applications that have the ability to interpret and use situational context to provide a basis for *adaptive* behaviour based upon changes in context.

There have been numerous attempts to define context and context-awareness and these have typically focussed almost exclusively on location. This class of application is termed *location-aware* and can be considered a sub-class of the more general area of context-aware applications. This section provides some background to the term context and context-awareness and references previous work where appropriate.

The work on Active Maps [Schilit,94a] by Schillit *et al* at Xerox PARC first introduced the concept of context-awareness. In their work they made use of location, identities of nearby people and objects as well as the changes that occur to those object over time as forms of context. Brown *et al* in their work on the Stick-e Note Architecture [Brown,96] describe the different forms of context that can be used by computers as: location, the presence of objects and people, temperature and blood pressure or more generally as any environmental factor that might influence the activities on a computer, provided it can be sensed. In Dey *et al* [Dey,00b], an abstract definition is provided in relation to their work on the context toolkit,

> *"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."*

Schmidt *et al* [Schmidt,98] in their work propose two general categories for context: human factors and physical environment each with three further subcategories. Human factors are structured into the following three subcategories:

- **User**: Is characterised by user habits, mental state or physiological characteristics.

- **Social Environment**: Is characterised by the proximity of other others by their social relationships and collaborative tasks.
- **Task**: Define goal directed activities or the general goals of the user.

The physical environment is structured into the following three subcategories :

- **Location**: This could be absolute, for example a GPS coordinate, or relative, for example inside a particular room.
- **Infrastructure**: Describe the surrounding computing and interaction environment.
- **Conditions**: These are physical conditions of the environment such as background noise, ambient light levels, brightness.

Schmidt *et al* also include time as orthogonal to these categories, since they argue that knowledge of previous contexts may prove useful to an application.

Finally, Dix *et al* [Dix,99b] in their paper define a taxonomy for context in terms of human-computer interaction and argue that a mobile device operates in a broad context that includes the network and computational infrastructure, the broader computational system, the application domain and the physical environment. The taxonomy can be summarised as follows :

Context	Relationship with	Issues
Infrastructure	network bandwidth, and reliability, display resolution	variability of service, user awareness of service, liveness of data
System	other devices, applications and users	distributed applications, pace of feedback and feed through, emergent behaviour
Domain	application domain, style of use, identification of user	situated interaction, personalisation, task and work studies, privacy
Physical	physical nature of device, environment, location	nature of mobility, location dependant information, use of environment sensors

Table 1.2 - Taxonomy for Context [Dix,99b]

Using this taxonomy it is possible to encapsulate all the aforementioned examples in this chapter within one of the defined contexts. The particular context types relevant to, and focussed on, in this thesis are :

- **Identity** - the identity of the user
- **Spatial** - location, orientation, velocity
- **Temporal** - time, date
- **Environmental** - light level, ambient noise, temperature, humidity
- **Social** - people in the vicinity
- **Resource** - nearby devices or services
- **Resource Management** - Resource usage or availability
- **Goals / tasks** - diary, document working on
- **User History** - previous user activity, interactions

1.5 Context-Awareness

Schilit and Theimer [Schilit,94b] defined context-aware as applications that are informed about context, or applications that adapt themselves to context. Later, Hull *et al* [Hull,97] and Pascoe *et al* [Pascoe,98a] define context-awareness more explicitly to be the ability of computing devices to detect and sense, interpret and respond to aspects of a user's local environment and computing devices themselves. More recently, Dey *et al* [Dey,00b] defines context-awareness as;

> *"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."*

The features of context-aware applications are generally categorised into one of the following categories: presentation of information and services to a user, automatic execution of a service, and, tagging of context to information for later retrieval

Presentation refers to applications that simply display contextual information or services to users, for example, an application displaying GPS coordinates to its user as she roams. Examples of context-aware applications that exhibit this behaviour include CyberGuide [Long,96], GUIDE [Davies,98a], Stick-e Notes [Brown,96] and Active Maps [Schilit,94]. However, Dey *et al* [Dey,00b] point out that it is often difficult to

distinguish between presentation of information and the presentation of services. For example, Schilit *et al* [Schilit,94a] explain that a list of printers organised by proximity to a user is an example of providing information to a user, however, this may depend upon its intended use. If a user merely looks at the list to become familiar with the names of nearby printers then she is using the list as information. However, if a user chooses to print to one of the printers in the list then she is using the list as a set of nearby services. Information and services are therefore treated in a similar manner to avoid attempting to make assumptions about the user's goal.

Auto execution includes context-triggered actions [Schilit,94b] and contextual adaption [Pascoe,98] and is the ability to execute or modify a service automatically based on the current context. Examples of context-aware applications that exhibit this behaviour can be found in GUIDE [Davies,98a], a tourist application capable of automatically displaying pages of tourist information as users approach particular attractions [Cheverst,01c], and onCue [onCue,00] which adds context-aware services to the standard Windows clipboard based on data copied to the clipboard.

Tagging of contextual information for later retrieval or contextual augmentation [Pascoe,98a] is the ability to associate digital data with a user's context. The forget-me-not [Lamming,94] project at Xerox Research Centre Europe (XRCE) (formerly EuroPARC) is a good example of an application that takes advantage of this type of behaviour. In forget-me-not Lamming *et al* use the idea that physical context can be a powerful cue for recalling past events, for example, when a user is trying to recall a particular document they were editing several weeks previously, they may not be able to remember the precise name of the document, but are more likely to remember that the document was presented at a particular meeting on a particular day with certain colleagues present.

1.6 Research Aims and Methodology

This thesis examines the issues and implications associated with supporting the development of mobile context-aware applications. More specifically, the thesis describes an investigation into the use of a context service based approach to support context-aware applications designed for use in mobile environments. The key research aims are therefore to:

- Investigate the practical issues relating to the development and deployment of context-aware applications designed for use in mobile environments.
- Capture a general set of requirements for supporting the development of context-aware applications.
- Develop, based on these requirements, a *context service* based architecture capable of supporting mobile context-aware applications.
- Develop a prototype implementation of the key elements of this architecture.
- Evaluate the effectiveness of the context service. Moreover, to ascertain its effectiveness and to examine the implications of this approach for a number of prototype context-aware applications.

The service based architecture introduces a separation of concerns between applications and the underlying infrastructure and one of the primary goals is to provide a flexible mechanism for experimenting and prototyping context-aware applications. In more detail, the hypothesis to be investigated is that a service based approach will afford the following advantages:

- **Flexibility**: To enable applications to adapt to use across a wide range of heterogeneous devices and execution environments independent of the underlying infrastructure or physical environment.
- **Evolution**: To aid application evolution and extensibility by using a separation of concerns between using context and the underlying infrastructure.
- **Partial Connectivity**: To enable context-aware applications to operate successfully in mobile environments even during periods of disconnected operation and to make use of discovery techniques to utilise new services.
- **Context Awareness**: To facilitate the awareness of users, devices and services across a range of heterogeneous devices and environments and to facilitate the sharing of context between application processes and instances.

Central to the research described in this thesis will be the implementation of a prototype context-aware application, the Lancaster GUIDE system [GUIDE,99]. The GUIDE system provides tourists to the city of Lancaster with an intelligent visitor guide able to present tourist information and services based on contextual factors such

as location, visitor interests, visitor history and a range of environmental factors. The ideas presented in this thesis have been established through the design, implementation and user evaluation of this system. Furthermore, this thesis demonstrates how the GUIDE prototype benefits from the service based approach in terms of operating in dynamic environments (flexibility) and through increased levels of awareness. The development of the GUIDE prototypes afford the following other significant contributions:

- A number of insights and detailed experiences relating to the development and deployment of a *real* and fully operational context-aware application.
- The development of a contextual information model capable of supporting geographic and context-sensitive information within a dynamic environment.
- An adaptive context-aware route guidance tool capable of providing personalised tours based on a wide range of extensible contextual attributes.
- The development of an adaptive mechanism for data dissemination to support a potentially large user community within close proximity.
- The initial results of a user field trial evaluation of the Lancaster GUIDE system based on a number of mobile devices with varying capabilities.

In order to validate the research hypothesis detailed above, this thesis utilises the Lancaster GUIDE system as the primary research vehicle. More precisely, an evaluation of the GUIDE system consisting of a retrospective analysis of the design process and a user evaluation field trial is used to determine how the GUIDE system operates within a mobile and dynamically changing execution environment. This, in addition to a critique of related work within the fields of mobile and context-aware computing, provides a mechanism for establishing a set of requirements that are able to form the basis for a context-service based architecture. The design and prototype implementation of a service based approach is realised before being evaluated using a modified version of the original GUIDE prototype which is able to make use of the properties afforded by the context-service architecture. The architectural evaluation then aims to validate the requirements and, more generally, the overall research aims in order to determine whether or not a context-service based approach provides a

flexible mechanism for developing, experimenting and prototyping context-aware applications designed for use within dynamic environments.

1.7 Thesis Outline

The thesis develops in the following way. Chapter two surveys systems support for mobility and the mechanisms currently adopted for supporting users within mobile environments. Additionally, current approaches to developing context-aware applications are detailed. The experiences of deploying a city-wide wireless network and the design and implementation of a prototype mobile context-aware application are described next in chapter three. In chapter four, these experiences are then analysed to inform a service oriented approach to context-aware application design. This analysis enables a set of requirements to be established for the design of an architecture designed for use in mobile environments. The implementation of a context service based architecture is detailed in chapter five. Chapter six revisits the requirements presented in chapter four and describes a number of motivating scenarios in order to evaluate the suitability of the service based approach. Chapter seven presents the conclusions of this thesis and identifies some areas for future work. In addition, appendix A describes the process involved in creating a user profile for use with the GUIDE system. Appendix B details the syntax and semantics relating to the GUIDE tags developed as part of the Lancaster GUIDE project. The development of an adaptive context-aware route guidance tool capable of providing personalised tours based on arbitrary contextual attributes is detailed in appendix C.

Chapter 2

Related Work

2.1 Introduction

The previous chapter described how advances in portable devices, wireless networking and sensing technologies have engendered the new paradigm of mobile context-aware computing [Forman,94]. This chapter lays the groundwork for the thesis and focuses on surveying systems support for mobility and the mechanisms currently adopted for supporting users on the move. More specifically, this chapter describes techniques for supporting user mobility, data dissemination of information to mobile users and current work relating to dynamic discovery. In addition, current approaches to developing context-aware applications are described. In particular, the ways in which these applications have been developed to make use of a wide range of contextual attributes are detailed. This chapter explores the specific problems which currently need to be addressed and argues that effective support for context-aware applications in mobile environments can only be achieved by building on the foundation of existing mobile computing concepts and context-aware architectures.

2.2 Supporting User Mobility

2.2.1 Introduction

Mobile computing research has evolved during the past decade from supporting disconnected operation and access to remote file systems [Satyanarayanan,90] to, more recently, multi-user, multi-device highly *adaptive* applications [HP,01], [Brumitt,00a] requiring access to heterogeneous communications infrastructures and sensing technologies.

In terms of supporting mobile users, research has typically focussed on two primary types of mobility. The first concerns only the mobility of the user, who for example, moves around an office building using fixed computing devices. An example of this is described by Ward *et al* [Ward,97] as part of their work on Active Bat infrastructure. The second involves the mobility of both user and computing device, for example, a field engineer using a pen-tablet and working on the move [Brown,98b], [Davies,94]. Although this thesis is primarily concerned with supporting the latter, it is pertinent to note the issues relating to both forms of mobility. The following sections detail the current approaches to supporting mobile users, including supporting disconnected operation, interaction within mobile environments and current resource discovery techniques.

2.2.2 Disconnected Operation

2.2.2.1 Mobile File Systems

Much of the early work on facilitating mobile information access was based around the notion of disconnected operation, that is, enabling existing applications to be used on hosts that could become disconnected from their home network due to user mobility, or network or server failures. This early research largely stemmed from work on distributed UNIX file systems such as the Network File System (NFS) and particularly the Andrew File System (AFS) [Satyanarayanan,85]. The initial aim of AFS was to develop a scalable and secure mechanism that enabled large numbers of users, spread across a number of remote hosts, to collaborate and share data. Following on from this, the Coda file system [Satyanarayanan,90] strived to integrate

the use of portable computers whilst remaining resilient to server and network failures.

The key mechanisms for supporting disconnected operation in Coda include hoarding (or user-assisted cache management), update logging and reintegration upon reconnection. In essence, while the client is connected to the servers (in the *hoarding* state, as shown in figure 2.1) the local cache is loaded with files that are prefetched from the user's working set either periodically or at the user's request. The files prefetched are determined by the user's hoard database file which contains file and directory paths together with associated priorities.

If the client disconnects from the network, the client temporarily becomes a replica site. The file system moves in to the *emulating* state where it optimistically allows operations on the files in the cache to continue as if the servers were still available [Kistler,91]. All the file operations are logged to a local update log.

When a server becomes available the client moves into the *write disconnected* state [Mummert,95]. Operations are still logged as when "emulating". However, the update log is reintegrated to bring the server replicas up-to-date. By replicating at the file level, Coda can transparently support pre-existing file-based applications.

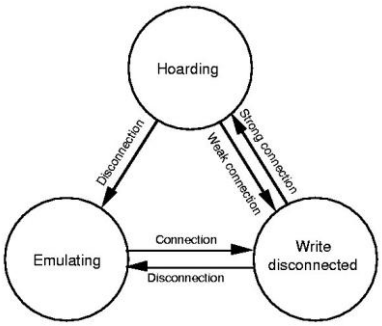

Figure 2.1 - The CODA state transition diagram

2.2.2.2 Bayou

Bayou [Demers,94] uses weak consistency replication techniques to manage replicas of shared calendars, electronic mail messages, databases, documents, and other artefacts that are central to collaboration. Rather than providing transparent support for an existing file system, Bayou focuses on supporting applications which are aware

that they are reading weakly consistent data and that their write operations are likely to conflict with others at some point. Bayou's design focuses on supporting application-specific conflict resolution mechanisms to ensure that replicas move towards eventual consistency.

A Bayou application's code and data are replicated at multiple clients and multiple servers. Bayou servers provide synchronisation services to the clients, and then synchronise among themselves. A client can synchronise with any server, with updates propagating from one server to another until they reach a designated primary server, at which time they become committed. Application developers may write conflict resolution procedures to resolve conflicts such as writes to the same database record. These procedures allow users to specify alternative updates in case primary updates conflict. Developers may also supply functions to detect conflicts specific to a particular application.

2.2.2.3 The Rover Toolkit

The Rover toolkit [Joseph,95] offers applications a uniform distributed object system based on a client/server architecture. Rover provides mobile communication support based on two concepts: Relocatable Dynamic Objects (RDOs) and Queued Remote Procedure Call (QRPC). A relocatable dynamic object is an object with a well defined interface that can be dynamically loaded into a client from a server (or vice versa) to reduce client-server communication requirements. Queued remote procedure call is a communication system that permits applications to continue to make non-blocking remote procedure calls even when a host is disconnected. Requests and responses are exchanged upon network reconnection. Updates to shared objects are permitted even when disconnected and consistency is ensured using application-level locking or by using application-specific algorithms to resolve uncoordinated updates to objects [Joseph,95].

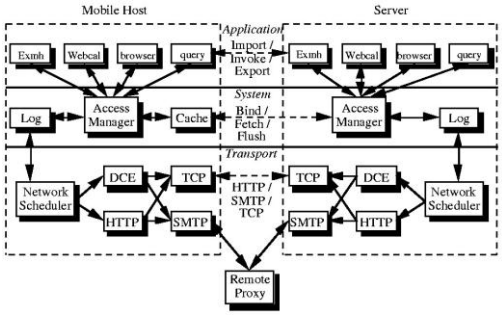

Figure 2.2 - The Rover architecture

2.2.2.4 TACOMA and TACOMA Lite

The notion of mobile objects or, more specifically, mobile code [Jacobsen,99] is taken further in the work on TACOMA [Johansen,97] and TACOMA Lite at the University of Tromsø which adds software mobility to mobile, handheld devices. TACOMA is based around the notion of folders and a meet operation. Folders enable an agent to transfer data from one agent to another. A folder stores code or data that the agent may access. A meet operation is called to enable a program to be started on a specific host by a program executing on that same or even a remote host and allows agents to communicate with each other. During meets agents are able to exchange folders. An example application based on TACOMA is the Stormcast weather warning system, which allows thin clients such as mobile phones to send short programs using SMS (short message service) messages [GSM,00].

Jacobsen and Johansen in their paper on TACOMA Lite [Jacobsen,97] describe how an infrastructure supporting mobile code is either stateful or stateless and the distinction being whether or not the system pre-empts and saves the low level state of a running process. A stateful system can pre-empt a running process, capture the entire state, transfer this over the network and restart the process at the destination host (analogous to process migration in distributed systems). Examples of stateful mobile code systems include Voyager [Voyager,01] or Agent-Tcl [Gray,96]. A stateless system does not capture the entire state of a running process but instead the code explicitly saves necessary state needed before migrating to another host. Furthermore, code is moved and activated at fixed, application specified entry-points, for example, a specific function with the code. This produces less complex and more portable systems but offers a trade-off against transparency.

2.2.2.5 Summary

This section has described methods of supporting user mobility and in particular facilitating information access during disconnected operation. The mobile file system approaches, which play an important role in supporting legacy applications, were presented first. These systems demonstrate effective file system use by weakening file consistency and offering optimistic access to cached files. Early solutions have often focussed on making user mobility transparent at the application layer and have been designed to allow legacy (i.e. mobile unaware) applications to run without modification in a mobile environment. In contrast, approaches like Bayou have been designed with mobility as a central concern and have been specifically designed to make potential inconsistencies visible to applications. The basic premise adopted by the mobile agent approach is that the processing of data items can be migrated to remote hosts and return only relevant data. Agent based techniques are similar to deploying a proxy server between a client and server to select the data on behalf of the client. However, unlike proxies which often reside at a fixed site, agents offer a more dynamic approach and typically form a more general architecture, enabling agents to dispatch and interact with other agents to accomplish their task.

2.2.3 Interaction Models

2.2.3.1 Overview

The mobile computing paradigm has posed a number of data management issues to the research community, in particular the problems of managing location dependant data, wireless data broadcast, disconnection management and energy efficient data access. The following section describes current approaches to data dissemination within mobile environments.

2.2.3.2 Traditional Data Dissemination Techniques

Franklin *et al* [Franklin,98] describe the following three main characteristics as useful when comparing traditional data dissemination:

- **Client Pull vs. Server Push:** Client pull refers to information transfer initiated by a client application, for example, a web browser making an explicit request

for a web resource from a remote server. Server push is server initiated and involves sending information to clients in advance of any explicit requests, for example, web based services such as the PointCast network [Pointcast,96], [Franklin,98] offering push based customised news alerts and information for a variety of topics including weather, sport and finance.

- **Periodic vs. Aperiodic:** Aperiodic data delivery can be applied to both push and pull. Aperiodic is analogous to event driven. For example, consider a client application responsible for displaying stock prices to its user; the transmission of information from the server to the client could be imitated by the user making an explicit request (pull) or due to some data items being updated (push). Periodic delivery is performed according to some specific schedule, e.g. hourly.
- **Unicast vs. 1-to-N:** Unicast communication allows a data item to be transferred from a server to a single client while 1-to-N communication allows multiple clients to receive the same data item simultaneously.

Using the characteristics described above, it is possible to classify some of the existing data delivery mechanisms accordingly.

	Client Pull	Server Push	Periodic	Aperiodic	Unicast	1-to-N
Request / Response	yes			yes	yes	yes
Polling	yes		yes		yes	yes
Publish / subscribe		yes		yes	yes	yes
Broadcast Disks		yes	yes			yes

Table 2.1 - Classification of data delivery mechanisms

2.2.3.3 Alternative Approaches to Data Dissemination

The following section describes alternative approaches to the traditional data dissemination techniques presented above.

Broadcast Disks

Broadcast Disks [Acharya,95a] developed at Brown University offers a periodic push mechanism aimed at supporting 1-to-N communications over wireless links. In

essence, the broadcast channel is regarded as a storage device and data is repeatedly and cyclically transmitted across the channel. The broadcast disk technique has two main components (as shown in figure 2.3). First, multiple broadcast programs (or "disks") with different latencies are superimposed on a single broadcast channel. Secondly, the technique integrates the use of client storage resources for caching and pre-fetching data that is delivered over the broadcast channel. These two mechanisms allow their approach to provide increased availability for critical data and improved performance for data access [Acharya,95b], [Imielinski,94].

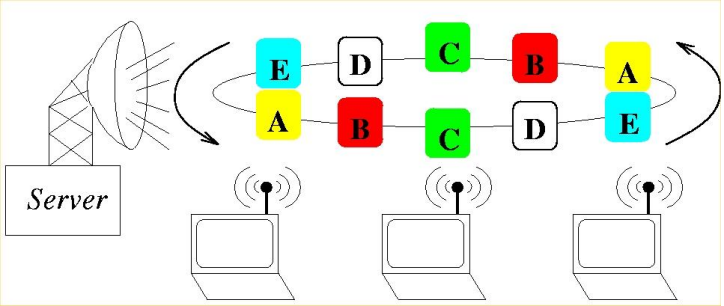

Figure 2.3 - Broadcast Disks

Linda (L^{2}imbo)

L^{2}imbo is a tuple space based distributed platform for mobile environments based around the original work on Linda [Friday,97] and the notion of *tuples* and *tuple spaces*. Tuples are typed data structures each consisting of a collection of typed data fields. Tuples exist within shared data objects called tuple spaces and tuples can be deposited, read and removed from the tuple space. Changes to a tuple can be made by withdrawing from the tuple space, amending and reinserting it, since tuples cannot be altered while resident in the tuple space. Tuple spaces are shared between collections of processes, all of which have access to the tuples within the tuple space. Friday *et al* [Friday,98] argue that the tuple space paradigm offers a number of attractions for mobile environments including the persistence of tuples, the guaranteed unique withdrawal of tuples and the fact that communicating processes producing and consuming tuples need not co-exist. This temporal decoupling allows processes to interact via the tuple space without the need for explicit synchronisation. In addition, communications are anonymous, that is, client and servers are unaware of each other's identity, although it is possible to encapsulate identity information into tuples to facilitate direct communications (as shown in figure 2.4). In [Wade,00] Wade

highlights some drawbacks of this approach for synchronous groupware and continuous media applications, primarily the slow enumeration of a tuple space, that is, to discover all group state all tuples must be removed from the tuple space. Furthermore, since there are no timeliness guarantees on the propagation of new tuples or relating to the recovery of missed state, it is difficult to establish whether a tuple exists within a given distributed tuple space since caches can be inconsistent.

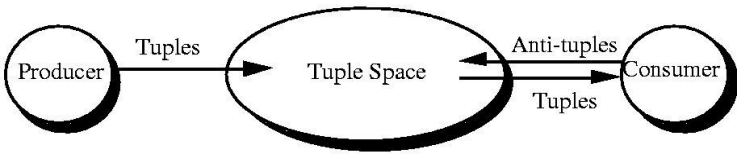

Figure 2.4 - Tuple Space communications

Cambridge Event Architecture

Spiteri *et al* describe an event based approach to supporting distributed *active* systems [Spiteri,98]. An event is defined as an asynchronous occurrence containing details of an activity that has occurred within a distributed system and also represents an index point into application sessions. The event based architecture described includes an even repository for capturing and storing events. Furthermore, to simulate the replay of activity sequences or for rebuilding lost state the architecture supports the injection of events back into distributed application components. Event-based systems allow any type of event or combination of events to be used to trigger further activity within an application, for example, automatically switching the lights on as a room is entered. The model is one of registration and notification and distributed clients may register their interest with event sources, which respond asynchronously upon detection of an event within the range of interest and notify interested parties. The functionality offered by the event repository has been demonstrated through a number of applications; monitoring and review of user activities, and tracking of user movements to be able to replay them in a virtual reality environment.

2.2.3.4 Summary

This section has focussed on the issue of data management and it has been shown how there are similarities between the event based systems and tuple space paradigm in terms of interaction models. More specifically, the generation of events can be

compared with the creation of tuples. However, application processes never directly interact under the tuple space paradigm. A more in-depth survey of these techniques can be found in [Wade,00] in which Wade argues that the inherent temporal and spatial de-coupling offered by the tuple space paradigm has advantages for the development of mobile adaptive applications.

2.2.4 Dynamic Discovery

2.2.4.1 Overview

The basic function of resource (or service) discovery is to allow users and applications to deploy, discover and interact with the services provided by the devices and software components within the network [Izadi,00a]. Although originally developed for zero configuration networks (e.g. domestic or office environments) there has been a recent trend towards using this technology for mobile and ubiquitous computing environments. Within these domains scenarios generally centre around enabling users to locate and utilise services within close proximity, such as entering a new building and discovering nearby printers from their PDA [Edwards,99].

A number of discovery mechanisms have recently emerged aimed at addressing the goal of simple, seamless and scalable device inter-operability. Examples include Microsoft's Universal Plug and Play (UPnP) [UPnP,00], Sun's Jini [Edwards,99] and the Service Location Protocol (SLP) [Guttman,99b]. All these technologies aim to provide the following basic capabilities:

- **Announcement:** Ability to announce its presence to the network.
- **Discovery:** Automatic discovery of nearby devices.
- **Description:** Ability to describe its capabilities (services) or query the capabilities of other devices.
- **Configuration:** Self configuration without administrative or user intervention.
- **Interoperability:** Seamless inter-operability with other devices.

The following section looks at each of the current discovery mechanisms in turn.

2.2.4.2 Universal Plug and Play (UPnP)

Universal Plug and Play (UPnP) [UPnP,00] aims to simplify the transparent interconnection of intelligent appliances, wireless devices and services by leveraging upon Internet technology and can be regarded, from a functionality point of view, as an extension of the concept of Plug and Play to support pervasive peer-to-peer network connectivity. In essence, the UPnP communication model is based around user controlled points, controlled devices and services (shown in figure 2.5). An entity, known as a Bridge, is used for representing devices that do not support the native UPnP controlled device functionality. The controlled device acts as a container for services and, possibly, other devices and typically represents some physical device, for example a VCR. A device may contain any number of services such as Play, Rewind, etc.

UPnP discovery is based on the Simple Service Discovery Protocol (SSDP) [Goland,99], a protocol designed with the goal of allowing devices to discover each other's presence within a network without configuration, management or administration. SSDP is based on HTTP over both unicast (HTTPU) and multicast (HTTPMU) UDP. When added to a network, a control point searches for devices and uses unique ids to identify them. As soon as a device is added to a network it advertises its services and periodically refreshes these advertisements. These advertisements contain a service type and a URL for the service being advertised. Devices must cancel advertisements when removed from the network and must respond to search requests from control points. All advertisements and search requests are based on IP multicast and search response message are unicast UDP which must contain a URI that identifies the resource (e.g. "cs:printer") and a URL to an XML file that provides a description of the announcing device [Goland,00]. This description includes, amongst other things, a URL that can be used to access the user interface for the device and a list of state variables associated with the service. For example, an object representing a printer might have a variable `auto-feed` with possible values on and off.

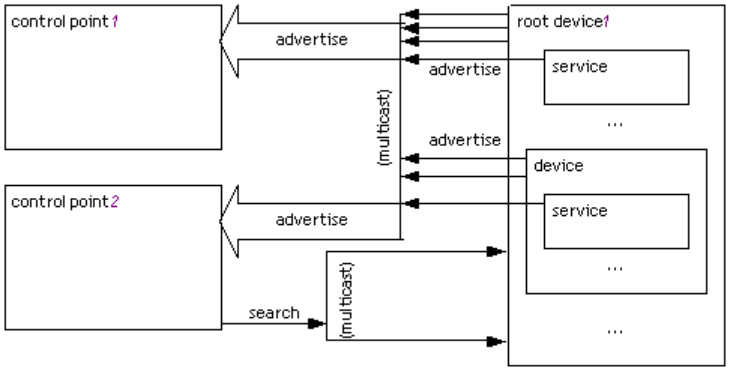

Figure 2.5 - Universal Plug and Play (UPnP) architecture [Microsoft,99]

2.2.4.3 Jini

The Jini technology from Sun Microsystems [Sun,99b], [Sun,99c] allows any network made up of services (printers, databases) and clients of those services to be managed without administration. Jini relies on the concept of a federation which is a collection of autonomous devices that can be made aware of each other's presence in order to cooperate with each other. To facilitate this, Jini includes a number of lookup services that maintain state information relating to devices available on the network.

Before a device can enter into a federation it must first discover a lookup service. Once a lookup service is located, a device can tell the service about itself (i.e. register) or the device can query service information relating to other devices in the federation. During the registering process attributes that are used to describe the service offered (name/value pairs) can be set which enable queries from other devices to be matched and following this devices upload Java byte code to the lookup service. This code acts as a "proxy object" that can be used to contact an interface on the device and invoke methods. This enables the querying of a device to automatically download a proxy and call methods, normally made using Java Remote Method Invocation (RMI), inside the device. The Jini lookup service can also be used by clients wishing to use a service. This involves searching the lookup service by specifying either its type, a unique identifier of the service or the attributes. The returned value is the proxy object from the service item and this contains methods to enable interaction between the service provider and the client. Once a proxy has been downloaded a client may communicate directly with the service as shown in figure 2.6 [Waldo,99].

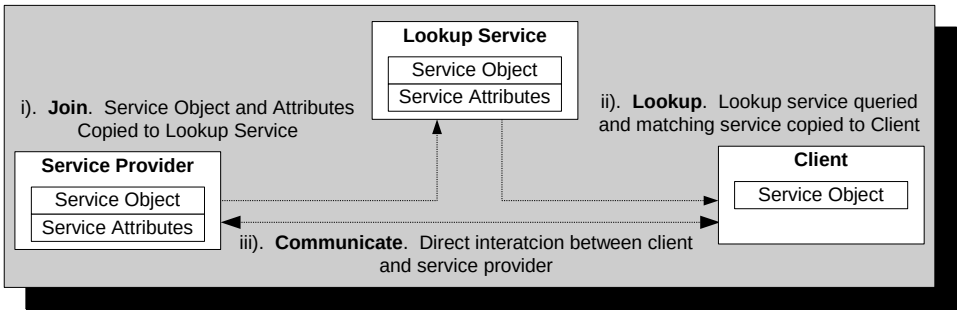

Figure 2.6 - Jini lookup service interactions

A key feature of Jini is that it is independent of the underlying transport protocol unlike UPnP and SLP which are based around TCP/IP. In addition, Jini supports the notion of events, so for example a digital camera could specify an interest in available printing services and once such a service becomes available a notification can be sent to the digital camera.

Jini lookup services make use of registration leases with expiry information to ensure that services have an accurate representation of the available devices. Devices are required to renew their leases periodically for the lookup services to maintain their registration information.

2.2.4.4 Service Location Protocol

The Service Location Protocol (SLP) [Guttman,99b] is a discovery mechanism aimed at enterprise networks with multiple shared services. SLP offers two alternative operating modes based around a directory agent, which is a repository for the available services.

The first operating mode involves the use of a directory agent by a user agent which allows a client to interact with it to discover services. More specifically, when a client generates a user agent this in turn generates a multicast request to obtain the location of a directory agent. The directory agent is also capable of periodically advertising its presence to user agents. Service agents acting on behalf of services wishing to advertise their presence discover directory agents in the same way as user agents. When present, a service agent registers with a directory agent and this registration is updated periodically.

The second mode is used when a user agent cannot locate a directory agent. In this scenario a user agent requests services directly by making explicit requests destined for a specific multicast group. Furthermore, user agents multicast service requests to a predefined specific multicast address which service agents respond to using unicast.

2.2.4.5 Summary

Although resource discovery is a key requirement for highly mobile environments, current discovery mechanisms generally assume fixed, high speed, reliable networks and, in addition, target home or corporate environments, an assumption explicitly presented in the Jini and SLP specifications. For use in mobile environments, issues such as interoperability and scalability alone currently pose significant challenges for application developers. Moreover, the presence of a diverse set of service discovery technologies available raises the issue of how to deal with the disparate range of device representations and interaction models. For example, UPnP defines XML schemes to represent devices, SLP uses URL syntax and Jini serialised Java objects. Clearly, applications capable of interacting with more than one of these discovery mechanisms simultaneously poses a great challenge for the application developer, particularly if the mobile device in question has limited resources, such as a PDA. Furthermore, in terms of scalability, as the number of clients and services within an environment increase, so the burden due to dynamic service discovery and interaction increases. In an analysis of current discovery mechanisms, Friday *et al* [Friday,01a] describe how a single client joining a network consisting of a single root device with a single sub-device and service type results in a total of 54 messages (datagrams) being generated. Thus, as the number of clients and services increase, low-bandwidth networks may become saturated with service announcement traffic. Table 2.2 summarise the key features of these architectures and a more in-depth comparison of these can be found in [Bettstetter,00], [Friday,01a] and [Jose,01a].

Feature	UPnP	Jini	SLP
Developer	Microsoft and various	Sun Microsystems	IETF
Network Transport	TCP/IP	independent	TCP/IP
Programming Language	independent	Java	independent
OS and Platform	independent	independent	dependent
Code Mobility	no	yes	no
Leasing Concept	yes	yes	yes
Security	IP dependent	Java based	IP dependant

Table 2.2 - Key features of discovery mechanisms

2.2.5 Analysis of Supporting User Mobility

The work described so far has introduced a number of different mechanisms for enabling users to achieve consistent levels of service despite user mobility. Mobile file systems such as CODA allow legacy applications to be supported although they do not adequately provide information to higher layers to allow applications to adapt to changes in user mobility. Using this additional information, mobile applications can apply a number of techniques to efficiently support mobility. Techniques such as filtering, agent based approaches like TACOMA, or service migration as found in Rover may all be used to help reduce bandwidth requirements. Data dissemination techniques such as Broadcast Disks and event based approaches such as the Cambridge Event Architecture may also be used to provide energy efficient and timely access to information respectively. However, there is a need for flexibility in terms of the data distribution techniques offered to developers of mobile applications, a factor presented by Schilit in his thesis [Schilit,95]. More recently, research has focussed on methods of supporting dynamic discovery of devices or services, although there are currently no unified or lightweight discovery solutions suitable for mobile environments. Furthermore, as shown in the next section, support for the discovery of more abstract services other than physical devices is not yet provided, for example, a user being notified that a work colleague has entered their locality.

2.3 Context-Awareness

2.3.1 Introduction

In order to achieve effective operation in mobile environments, mobile applications must adapt in response to changes in their environment [Davies,94], [Katz,94]. These *adaptive applications* in general require information relating to changes in their surrounding environment, in particular their network quality of service (QoS). To achieve this, aspects of the environment must be sensed or discovered and disseminated among interested parties accordingly. Noble *et al* [Noble,97] describe a taxonomy for adaptation which ranges from placing the responsibility of dealing with adaptation on an individual's application (*laissez-faire*) to placing the responsibility for adaptation on the system (*application transparent*). Between these lies *application-aware* adaptation which is based on a collaborative partnership between application and the underlying system. The latter approach allows applications to determine how and when best to adapt, but preserves the ability of the system to monitor resources and enforce allocation decisions. Satyanarayanan *et al* [Satyanarayanan,90] uses Coda as a research vehicle into application-transparency adaptation and more recently have focused on application-aware adaptation with the Odyssey platform [Noble,99]. Adaptive applications have traditionally focused on the monitoring of the network QoS as means for triggering adaptation. The following section describes work in the field of context-aware application design, a field which is more general than mobile adaptive computing since all aspects of the operating environment may be monitored and used in order to trigger application adaptation.

Schilit [Schilit,94b] in his work on the PARCTAB system at Xerox PARC [Want,95] describes the context-aware life cycle as *context discovery* (content capture or sensing), *context selection* (interpretation) and *context use*. This broad structure will be used to analyse related work in the area of context-aware computing.

2.3.2 Context Capture

2.3.2.1 Introduction

Brown *et al* [Brown,00b] describe several mechanisms for context capture. Firstly, context can be captured from the environment through the use of *sensors* such as GPS

receivers [Dana,98] or via the state of a user's equipment, such as battery life. In addition, existing information or applications such as diaries, 'to do' lists or weather services can be used to obtain context. Context may also be captured from *user* and *task models*, for example, the fact that a user is vegetarian could be considered valuable context when specifying the task "Where is the nearest café?" Context may also be gained *explicitly* by the user specifying their current interests or needs [Pascoe,98b], for example, "I am hungry and tired". Finally, context can also combine aspects of the physical and virtual worlds, for example in the Lancaster GUIDE system described in chapter three, users are able to request information relating to their location virtually within the information space while at the same time having the information displayed and tailored to aspects of the physical world.

2.3.2.2 Active Maps

Work on the Active Badge system [Want,92] at Cambridge AT&T Labs (formerly Olivetti Research Laboratory (ORL)) formed the starting point for much of the work on context-aware computing. Active Badges are small devices worn by personnel that transmit a unique identifier every 10 seconds using infra-red. A network of ceiling based sensors receive these transmissions and are able to determine the location of the badge, and therefore its wearer within the building. Active Badges have been successfully deployed within a number of academic institutions and applications based on this technology include Active Map [Want,92] and FLUMP (The FLexible Ubiquitous Monitor Project) [Finney,96a]. The Active Map annotates graphical floor plans of the AT&T labs with location information gathered from people and objects, such as printers, allowing users to determine the required service. FLUMP, based at Lancaster University, enables useful information (such as e-mail and diary information) to *follow* people around the Computing department using a series of wall mounted monitors distributed throughout the building. These monitors make use of WWW technologies to render information and the migration of information between monitors is triggered by Active Badge sightings. Although their relative small size and weight make the Active Badges convenient to wear, they do not offer the fine-grained 3D location and orientation accuracy required to track people with precision within indoor environments for applications such as the intelligent videophone system. This system utilises a number of cameras placed around a room and

continuously selects a camera view with which the user's face can be seen. This allows a user to wander freely around an office room during a videophone conversation. To achieve this higher level of positional accuracy, Harter *et al* [Harter,99] have deployed a low-power, wireless and relatively inexpensive ultra-sonic location system known as Active Bats (bat). This approach uses a series of ceiling based transceivers to monitor people and objects within the building wearing a Bat. The use of Active Bats overcomes some of the limitations of the preceding Active Badge infrastructure and offers finer grained location in 3 dimensions. However, due to the infrastructure requirements to gain this information (deployment of a large number of ceiling based transceivers) it is unlikely to be adopted as the ubiquitous indoor location technology. Consequently, a similar although more lightweight approach to fine grained 3D positioning within indoor environments is currently the focus of research by Randell *et al* [Randell,00] as part of their work on the CyberJacket.

2.3.2.3 Cyberguide

Another example of a location based context aware application is Cyberguide [Long,96], an indoor mobile tour guide for visitors of the GVU (Graphics, Visualisation and Usability) Centre at Georgia Tech. Visitors carrying Apple MessagePads have information presented to them based on location and orientation information gathered from a series of ceiling based Infra-red sensors. The original Cyberguide implementation suffered from a number of restrictions which included a tight coupling of the positional information and communications infrastructure and the use of static maps of the environment. Indeed, the ceiling based sensors used to provide location and orientation information were also employed to provide the communications ability so changes in sensor positions involved parts of the application to be re-written as well as the reloading of the new information model. A further limitation is the use of a hard wired infrared positioning system based around a series of remote controls suspended from the ceiling, each with a different button taped down to provide the unique infra red beacon. The use of static configurations therefore has a detrimental affect on the evolution and extensibility of the Cyberguide application [Dey,01].

2.3.2.4 Teleporting

The Teleporting System developed at AT&T Labs is a tool for experiencing mobile X sessions. Teleporting [Bennett,94] enables the mobility of X-windows user interfaces by allowing them to relocate between displays. Teleporting makes use of location, identity and activity to enable a user to materialise and de-materialise their desktop to and from a display using simple commands. However, teleporting relies on the X-windows windowing paradigm although some of the mechanisms adopted in their approach could be applied to other applications. An example of this is the use of a *proxy X server* which is used to relay communication between users and real displays (servers). When display relocation is required, the proxy server transparently manages the handover of the user interface between displays. The use of transparent proxies is a mechanism to support user mobility (disconnection) adopted by other work such as Top Gun Wingman [Fox,98].

2.3.2.5 CyberDesk

The CyberDesk architecture was built to automatically integrate web-based services based on context relating to user activity. In more detail, the CyberDesk architecture uses contextual information to dynamically integrate software modules, for example, a user highlighting a particular appointment with a work colleague from their diary will have a number of services suggested to them. The system is able to interpret the appointment and extract the relevant context in order to provide context-aware services, for example, offering to search for the selected text using a web-based search engine, look up the name in their address book or to send an email to that person. However, the system is very limited in the types of context it can handle and does not support multiple simultaneous applications. Finally, the context used by the system is not stored nor is it queryable. These shortfalls were identified when Dey *et al* used their approach to build an intelligent environment application [Dey,97] and some of these limitations were later addressed as work on the Context Toolkit described in section 2.3.3.2.

2.3.2.6 Summary

Current approaches to context-sensing centre around the use of location as the primary context type. Furthermore, mechanisms for managing this information are

simple and typically designed for particular location technologies and with specific target applications in mind [Bennett,97]. This tendency for utilising context in such a bespoke manner makes application evolution difficult and does not afford a trivial mechanism for transferring a system to an alternative environment [Long,96], [Dey,98]. More general approaches to the representation and use of context are required to enable application developers to offer re-usable components for use within future mobile context-aware applications.

2.3.3 Context Selection

2.3.3.1 Introduction

The following sections describe previous work relating to context selection, which include mechanisms for notification of contextual updates, context storage and context interpretation where interpretation refers to the transformation of one or more types of context into another type of context.

2.3.3.2 Context Toolkit

The Context Toolkit [Dey,99b] developed at Georgia Tech's GVU aims to separate the context acquisition process from the delivery and use of context. The Context Toolkit supports the acquisition and delivery of context using three types of abstractions: widgets, servers, and interpreters, as shown in figure 2.7.

- **Context widgets** are software components that provide applications with access to context sensed from their operating environment. They free applications from the context acquisition process by hiding the complexity of the sensors from applications. Each widget encapsulates state and a set of event callbacks. The state is comprised of contextual information that applications can exploit via polling or subscribing. Callbacks represent the types of events that the widget can use to notify subscribing applications. The widget also maintains contextual state allowing other components to retrieve historical context information.
- **Context servers** are used to collect the entire context about a particular entity, such as a person. The context server is responsible for subscribing to every

widget of interest and acts as a proxy to the application, collecting information for that particularly entity. The context server can be seen as a compound widget. As such, it has attributes and callbacks, it can be subscribed to and polled, and its history can be retrieved.

- **Context interpreters** are responsible for implementing the interpretation of context information. They transform between different representation formats or merge different context information to provide new representations.

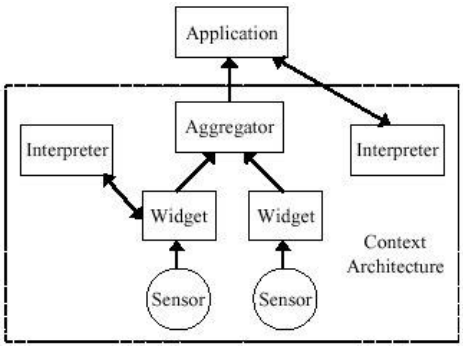

Figure 2.7 - The Context Toolkit architecture

Within this model, context widgets derive fine-grained context information, which may be interpreted and presented to context servers. For applications to receive notifications about a particular type of context such as the activity within a specific location, they would have to subscribe for events from the actual context widget. However, if events about particular system entities, such as users, are required then the application must register for notifications with the context server. Although the Context Toolkit includes mechanisms for enabling applications to register for call backs there is no inherent support for user mobility. Consequently, should an application register for updates (events) for a particular type of context and then become disconnected from the network due to host mobility, there are no mechanisms for synchronisation of state once the network connected is re-established.

2.3.3.3 Context Information Service

The Context Information Service (CIS) [Pascoe,98b] is a further architecture which supports context-aware applications through the maintenance of an object-oriented view of the world comprised of *artefacts, states, sensors, synthesizers, monitors* and *catalogs*. Artefacts have a name, type and a set of contextual states associated with

them. For example, the world may contain a person artefact named 'Keith' with a location state. CIS clients can access the state of any artefact in the world (e.g. Keith's location). The CIS also consists of a *state catalog* and an *artefact catalog* which provide templates for creating new artefacts and interactions between them.

Sensor arrays are used to collect contextual data (e.g. location from a GPS device) and synthesizers are used to synthesize or aggregate contextual data from other artefact states, for example synthesizing a sunset from a location, weather and a time state). Contextual data from sensors or synthesizers is directed to appropriate artefacts under the control of a monitor, which provides strict quality of service guarantees, for example, location must be accurate to within 50 metres.

The contextual model constructed using these components is then made accessible through four CIS service components (shown in figure 2.8), which act as the interface through which any client program or user may construct, view, or manipulate a shared contextual model of the world. This approach is currently work in progress and no implementation details are currently available.

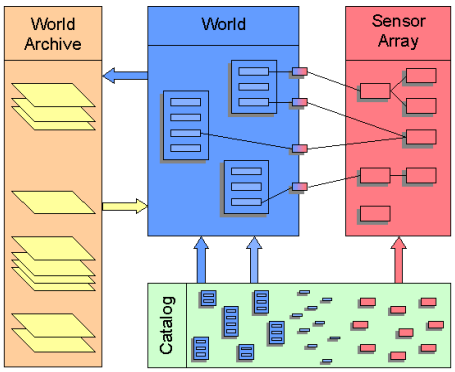

Figure 2.8 - The CIS service components

2.3.3.4 TEA

The Technology for Enabling Awareness (TEA) [Schmidt,98b] approach to supporting context awareness involves transforming sensor readings into context profiles, for example, transforming sensors values such as light = 90% and noise = 75% into context such as "in a meeting". This is achieved through a four layer architecture and primarily through the use of *sensors*, *cues* and *context profiles*. Output from sensors is regarded as low level and therefore requires transformations or

filters to be applied before being of any significant value to applications. For example, the output from a light sensor could be replaced with mean and variance values. Filters and transformation of raw sensor data are examples of cues (layer 2) and these can be used to give a better interpretation of the data. The third layer involves the mapping of cues to context profiles specified by the user and this forms the focus of the TEA approach. This mapping layer must be able to transparently adjust itself to contexts that the user will visit and furthermore, this will enable the fourth layer to utilise the context information in order to adapt the behaviour of applications or devices. This architecture was used to determine the state of a mobile phone in order to automatically configure its profile. For example, a user walking in an outdoor environment will have the cell phone configured so that the volume is set to maximum and the vibrating alert turned on. However, if a user is thought to be in a meeting then a silent mode may be selected which re-directs all phone calls to voice mail. The TEA layered architecture is summarised in figure 2.9.

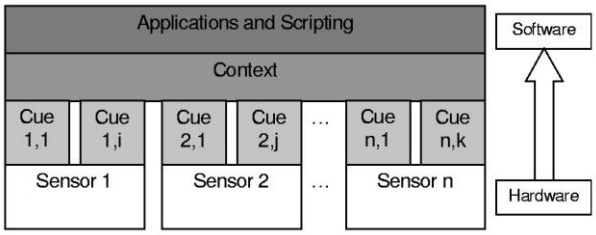

Figure 2.9 - The TEA layered architecture

2.3.3.5 Audio Aura

The goals of Audio Aura system [Mynatt,98] are to provide serendipitous information related to people's physical actions in the work place via background auditory cues and to explore the use of audio to connect a person's activities in the physical world with information gathered from the virtual world. The system is based around a centralised server and relatively thin clients with most of the computation occurring on the server. The server maintains a history by storing context (location, identity and time) gathered from the environment, primarily through the use of Active Badges [AT&T,00]. The server also provides a notification mechanism which allows clients to stipulate their notification constraints although this requires advance knowledge of how the context is stored on the server, therefore reducing the separation between context acquisition and context use.

2.3.3.6 AROMA

The AROMA project attempts to provide peripheral awareness of geographically dispersed people who would like to stay in touch [Pederson,97], for example, allowing a person about to make a phone call to be able to see what is happening at the callee's site before making the phone call. AROMA allows auditory and visual data captured in a colleague's space to be delivered to another colleague's space and rendered in a variety of ways. The aim of AROMA is to explore the use of *abstract representations* of captured data as presence indicators. Its object-oriented architecture uses *capture objects* to encapsulate sensors and *abstractor objects* to extract features. Synthesizers take abstract awareness information and display it.

2.3.3.7 Summary

A tight coupling generally exists between the sensing and use of context [Brown,96][Finney,96b], although in the original call forwarding Active Badge application servers were placed between the application and the sensors to abstract the details of the sensing from the application. Furthermore, sensed context is generally low level, such as longitudinal or latitudinal coordinates provided by a GPS compass, and context interpretation may be required in order to provide useful information to applications. The work presented in the previous section describes current approaches to context selection and, in particular, methods of abstracting over the sensed context. Dey *et al* describe an architecture to support the building, execution and evolution of context-aware applications via the use of context abstraction. In contrast to the approach adopted by Dey *et al*, the Context Information Service (CIS) promotes a tight coupling between the application and the underlying sensors, taking an application-dependant approach to application development. Typically, the systems discussed thus far are limited to a portion of the context selection process, that is, either including mechanisms for notification of contextual updates or techniques for context abstraction and all omit implicit support for user mobility.

2.3.4 Using Context

2.3.4.1 Stick-E Notes

Brown [Brown,96] in his work on the Stick-e Notes framework proposes an architecture for supporting application designers in using context to perform context-aware behaviour with the goal of enabling non-programmers to author context-aware services. This architecture is based around the *stick-e note* metaphor which is the electronic version of a Post-It note and represents the association of context to objects. Stick-e notes can be attached to a range of contexts such as specific locations, people or objects and may also contain actions or rules that specify the type of behaviour which should take place when a particular context is entered. For example, attaching a stick-e note to a particular location such as a library and to a particular person can be used to trigger an event should that context be entered, i.e. user meeting that person in the library. A tour guide application was developed to demonstrate their approach and some of the ideas presented in this research formed the basis for work on the Lancaster GUIDE system described in chapter three.

2.3.4.2 PARCTAB and Active Maps

The PARCTAB system developed at the Computer Science Lab (CSL) at Xerox PARC [Want,95] integrates palm-sized mobile computers into an office environment (network) and served as a testbed for research in to the ubiquitous computing vision first described by Weiser [Weiser,91]. The Tab is a personal digital assistant (PDA) that communicates via infrared (IR) to a network of IR transceivers. The system is designed for indoor use in such a way that each room serves as a communication cell. As the tabs move from cell to cell (room to room) the underlying infrastructure provides an uninterrupted and reliable service. A key feature of the approach adopted by Schilit *et al* [Schilit,94a] is that most of the applications run on remote hosts and this places a high dependency on reliable communications throughout the IR network.

Within the PARCTAB system there exists three types of software component: *gateways*, *agents*, and *applications*. Gateways are responsible for sending and receiving data packets using IR signals. Each tab is represented by an agent which is responsible for tracking its location and also provides location independent remote procedure calls [Want,95]. Applications are built using a library of widgets designed

to accommodate the low IR-communication bandwidth and small display area of the tab. The PARCTAB system architecture is summarized in figure 2.10.

Schilit in his thesis [Schilit,95] presents an architecture to support the design of mobile context-aware applications. More specifically, in his research the architecture is based around three main components: device agents, user agents and active maps. Device Agents maintain the status and capabilities of devices, whilst user agents maintain user preferences and active maps maintain location information of devices and users.

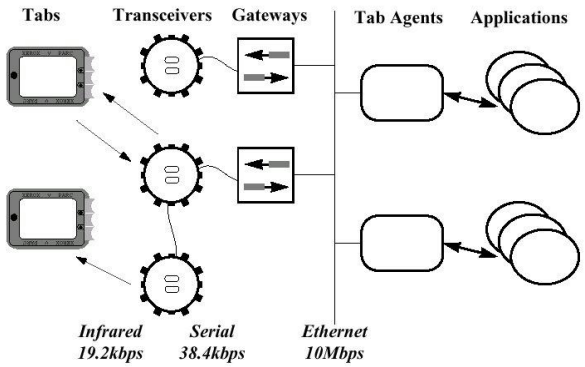

Figure 2.10 - The PARCTAB system architecture

2.3.4.3 Cooltown

Cooltown is an infrastructure to support "Web presence" for people, places and things [Kindberg,00], [HP,01] by associating each real world object with a web resource [Caswell,00]. Each web page is used to store information about the real-world object it represents and is automatically correlated with their physical presence. Cooltown primarily supports the display of context and services to its users, for example a user carrying a PDA walking through a city electronically picks up URLs for pages about the city itself and the places travelled through (e.g. a railway station, a shopping district, a café) and these are presented as web links.

The main component of the Cooltown architecture is a place manager that maintains a directory with the description of the people and objects physically present in that place. As people and mobile devices move, the state of the directory is updated to represent the current state of the place at any point in time.

Cooltown offers a discovery mechanism and abstraction components such as URLs for sensed information and web pages for real-world objects;F however their approach does not seem to offer support for low-level sensed information or support for other features of context-aware applications such as automatic execution.

2.3.4.4 EasyLiving

The EasyLiving project [Brummitt,00a] developed at Microsoft Research focuses on the development of an architecture and supporting technologies for intelligent environments and in particular the support of a coherent user experience via the aggregation of a diverse set of devices. The EasyLiving architecture encapsulates hardware device control, internal device logic and user interface presentation as service abstractions. These abstract service descriptions register with a lookup service and allow each service to expose a set of attributes so that other services may interact with them automatically. A major focus of the work is aimed at geometric world modelling [Brummitt,00b] and supporting geometric knowledge, that is, representing knowledge about the physical relationships between people, devices, places and things that can be used to dynamically assemble a set of UI devices for a particular interaction. Brummitt *et al's* approach uses software components to decouple the sensors from the application, and to provide transparent communications and discovery mechanisms to enable the availability of components to multiple applications. Communication between components occurs via an asynchronous message passing and an XML-based message protocol (SOAP [Box,00]). As part of the EasyLiving project an intelligent space has been created to demonstrate a number of applications, which include the teleporting of desktops among available displays, personalised media services which allow audio or video playback across a number of devices (such as CD, MP3 , DVD or Videotape) based on user location and interests [Meyers,00], and dynamic room controllers that provide context sensitive access to services based on co-location [Shafer,00]. EasyLiving includes mechanisms for discovery but does not provide explicit support for context interpretation or context storage for later retrieval.

2.3.4.5 Forget-Me-Not and Satchel

The work on Forget-Me-Not and the memory prosthesis [Lamming,94] at Xerox Research Centre Europe (formerly EuroPARC) aims to help with everyday problems such as finding documents and remembering names and addresses. Forget-Me-Not is an example of a context-aware retrieval system based on user activity (or context) such as phone calls made, emails received and rooms visited. As a user interacts with devices information such as the device name, location, salient details relating to operations performed and a timestamp are logged into a personal (private) biography. Similarly, the system logs the contextual features of encounters with other users, for example "John met with Sam in the reception". The biography therefore contains a sequence of encounters which may be browsed or filtered based on particular entities or activities and displayed to a user's PARCTAB screen [Schilit,95]. For example, a user could manipulate their PARCTAB biography to answer questions such as "Which document did I print just before meeting John in his office"?

Following on from earlier work on Forget-Me-Not, the Satchel project [Flynn,00] aims to take advantage of the user's context to extend the range of services available, in particular, to enable printing and transferring documents between users from the PARCTAB. Early prototypes allowed users to transfer documents to other users or printers by dragging and dropping document icons over user or printer icons. Subsequent prototypes have adopted a web browser metaphor, web standards and a DO-IT command, which are able to invoke the most appropriate action for any given context. A user's location determines the action of the system, thus for example, if a user selects a document when within close proximity to a printer, the document will be printed, however if a user is in closer proximity to a large display the document will be displayed instead. Other related examples of applications which strive to support their user in everyday tasks include the Remembrance Agent [Rhodes,96], Factoid [Factoid,01] and the context-aware WAP phone [Schmidt,00].

2.3.4.6 Summary

The preceding section presented related work in the area of context use which, as introduced in section 1.5, can be divided into applications that merely *present* contextual information to users, applications that are able to *adapt* or automatically

execute services based on context and applications that tag/store context for later *retrieval*. This author argues that context-aware applications must be able to support a flexible approach to context use in contrast to existing applications which are typically limited to dealing with only a portion of the available mechanisms and do not adequately support all three. Furthermore, the extent to which applications make use of context is currently a limiting factor and little re-use of context between systems and applications exists.

2.3.5 Analysis of Supporting Context-Awareness

This section has considered work related to context-aware application design, particularly in relation to the context-aware life cycle: context discovery, selection and use.

In general, application specific approaches to context-aware systems design has been adopted resulting in solutions that are inherently inflexible to application evolution. In more detail, by creating applications based upon specific technologies context-aware systems do not have widespread applicability since they are limited by the characteristics of the specific technology upon which they are based and, therefore, only available to systems supporting the particular technologies employed.

Furthermore, the approaches presented have not been intended to support the *sharing* of context between context-aware applications and most researchers have focussed on using contexts within a specific application domain, e.g. tourism, office assistants, and the types of context used within an application remain largely static. There is considerable argument to suggest that a number of valuable benefits exist for context sharing between applications such as Forget-Me-Not and CybreMinder [Dey,00a]. However, to achieve this successfully a separation of the representation of context is also required, in addition to insulating the context acquisition process from applications. In terms of context acquisition, current approaches, with the exception of the Context Toolkit [Dey,99c], suffer from a tight coupling between the application and the underlying sensor technologies used to acquire context. Furthermore, location has often formed the basis of context-aware application design and has most commonly been demonstrated through indoor environments [Abowd,97], [Bederson,95], [Fels,98], [Jose,99], [Oppermann,99a]. Clearly, other

aspects of the physical and computational environment can be utilised and work on teleporting, Cyberdesk [Dey,98] and TEA have exploited both identity and activity as forms of context in addition to location. Knowledge pertaining to time, history or people other than the user are still not widely incorporated into context-aware applications yet these are fundamental pieces of personal information used in everyday life [Abowd,99].

As described in this chapter, approaches to context interpretation including context aggregation [Dey,99c], context synthesis [Pascoe,98a] and context fusion [Schmidt,98b], [Chen,99] strive for a similar goal, transforming lower level (sensed) context into higher level context usable by applications. However, they offer alternative, and sometimes conflicting, approaches with varied support for notification mechanisms such that none seems suitable for use in highly mobile environments.

Based on the work described in this section it can be argued that existing approaches often focus on fixed infrastructures and utilise fixed information models. This suggests a strong requirement for a more general information model capable of utilising and sharing a wide range of contexts (location, identity, history, activity) and manipulating context in any number of the methods presented (display, adaptation, storage).

In summary, by developing systems with the above properties, it is possible to create a research environment in which issues relating to context-awareness can be explored more thoroughly. The rapid deployment of applications across a wide range of environments increases the opportunity to experiment within real world settings and thus, as a result, provide a sound basis for experimentation and deriving requirements for future generations of mobile systems and applications.

2.4 Conclusions

This chapter has presented a comprehensive survey of application level support for mobile context-aware applications. Current research tends to offer ad hoc, application specific mechanisms to context-aware application design and, in general, often neglects valuable techniques from the mobile computing field of research, in

particular, those which relate to user mobility, disconnected operation and resource discovery.

Based on this survey it can be argued that mobility and its consequences should not be masked from the developer of mobile context-aware applications. Moreover, one of the aims of this thesis is to identify new mechanisms for increasing the level of awareness. More precisely, the mechanisms that enable the level of awareness to be extended with respect to a user carrying a mobile device are investigated. This should encompass more than just the surrounding environment, such as locally sensed context, to include context pertaining to remote users and perhaps even their device capabilities. To facilitate collaboration among users in these highly mobile environments increased levels of awareness will be required to enable users to collaborate using a wide range of devices and this can only be achieved by fusing concepts relating to mobile and context-aware application design.

The remainder of this thesis focuses on the development of an architecture to support context-aware applications in highly mobile environments. First, however, chapter three goes on to describe the Lancaster GUIDE system, a mobile testbed used for research into issues relating to context-aware computing in dynamic environments.

Chapter 3

The GUIDE Prototype

3.1 Introduction

This chapter details the design and implementation of a prototype mobile context-aware application aimed for use in an outdoor city environment. This application was built to satisfy the requirements of tourists to the city of Lancaster and to provide them with an intelligent visitor guide [Davies,98a]. The chapter provides a number of insights into the challenges involved in building and deploying context-aware applications designed for use in mobile environments [Cheverst,00a], [Cheverst,00b]. The chapter opens with an overview of the application domain and describes the requirements capture process carried out early in the project to determine the real requirements of tourists. The main focus is then directed toward a detailed account of the design and implementation of the GUIDE prototype application developed to support city visitors.

3.2 The GUIDE Project Overview

3.2.1 Background

The Lancaster GUIDE project was initially a two year collaborative venture involving Lancaster University and Lancaster City Council and funded by the EPSRC (grant

number GR/L05280). The aim of the project was to investigate the provision of context-aware mobile multimedia computing support for city visitors, in particular, investigating the many issues and challenges that arise from the development and *actual deployment* of a context-aware electronic tourist guide in a practical real-world environment, i.e. the city of Lancaster. In more detail, the project aim was to develop a number of hand-portable multimedia end-systems which provide information to visitors as they navigate an appropriately networked city. The end-systems were designed to be context-aware, i.e. end-systems should have knowledge of their users and of their environment including, most importantly, their physical location. This information should be used to tailor the system's behaviour in order to provide users with an intelligent visitor guide [Mitchell,98].

3.2.2 General Requirements for Supporting City Visitors

An initial set of requirements for the GUIDE prototype were obtained through a series of semi-structured, one-to-one interviews with members of staff at Lancaster's Tourist Information Centre (TIC). In addition, several days were spent at the TIC observing the information needs of visitors, how the TIC staff supported those needs and in particular focussing on any potential areas for future improvement. The requirements identified were as follows [Davies,98a].

Ability to Support the Mobility of Tourists

One of the key requirements for GUIDE was the need to take into account the mobility of tourists. More specifically, tourists generally expect to, and are happy to, spend the day exploring the city and potentially walking for long distances. They may also be carrying with them items such as back-packs, cameras and other accessories. This means that the prototype application should be designed around a highly portable end-system, such as a portable PC or pen tablet. This device should be lightweight, able to operate outdoors in a variety of weather conditions and have sufficient battery capacity to sustain continuous or intermittent use throughout the course of a day.

Ability to Handle Multimedia Information

The prototype application must be capable of handling multimedia information. Tourists may wish to read textual information relating to a particular attraction but may also require the use of images, maps, animation, video or sound. Therefore the system must be flexible enough to make use of a rich collection of media types.

Flexibility

One of the key requirements for GUIDE was the need to provide sufficient flexibility to enable visitors to explore and learn about the city in their own way. For example, some visitors prefer to follow a guided tour while others may choose to explore on their own, following one or more guidebooks or street maps. Therefore, the system should be capable of acting as an intelligent tour guide or as a richly featured guidebook depending on the needs of the visitor.

It is also important that GUIDE enables visitors to control their pace of interaction with the system. For example, visitors should be able to interrupt a tour in order to take a coffee break whenever they desire. In addition, a visitor should not feel overly pressured by the system to leave an attraction prematurely.

Context-Aware Information

A further requirement was that information presented to visitors should be tailored to their context. Initially, two classes of context were identified, namely personal and environmental context. Perhaps the most significant piece of personal context is the visitor's interests, for example, in the city's history, maritime or architecture. Other examples of personal context identified include: the visitor's current location, any budget constraints, disabilities or, refreshment preferences they might have (e.g. vegetarian). Examples of environmental context include: the time of day, the opening times of attractions and the current weather forecast.

Context should also be used when presenting information to visitors, for example, information should be presented in a way that is suitable given context such as a user's age, technical background, or preferred reading language (e.g. English or German). Context should also be used to adapt the presentation of information, for

example, when visitors make return visits to landmarks, information presented should reflect this fact, e.g. by welcoming the visitor back. Oberlander *et al* [Oberlander,97] use the term *coherence* to describe the notion of tailoring the presentation of information based on what the user has already seen.

Support For Dynamic Information

During the requirements study we found there to be a significant requirement for the support of dynamic information. Such information should be made available to visitors whenever their context deems this to be appropriate. For example, consider the hypothetical scenario in which a visitor touring the city has expressed a particular interest in Lancaster castle. However, when starting their tour, the castle happened to be closed to the public because the courtroom, situated within the castle, was in session. As it transpires, the court session finishes early and so the visitor should be notified that the castle is now open to the public.

Support for Interactive Services

Studying tourist activities in Lancaster revealed that a surprising number of visitors make repeat visits to the TIC, often during the course of a single day. In most cases this is in order to ask members of staff specific questions or to make use of the services offered by the TIC, most commonly the booking of travel or accommodation. In order to help alleviate the need for visitors to walk back to the TIC to ask questions, the system is required to support remote communications. In addition, the system should also enable visitors to make accommodation bookings without having to return to the TIC. More specifically, the electronic nature of GUIDE should enable the system to offer greater flexibility than conventional pre-printed information sheets and the use of a network-based architecture should enable the system to keep visitors up-to-date with dynamic information and offer interactive services such as accommodation booking.

3.3 The GUIDE Infrastructure

This section details the design of the GUIDE system and outlines the hardware and software technologies adopted to underpin the application, associated constraints, and

a justification for these design choices. In particular, issues relating to the choice of end system selected, the underlying infrastructure and the prototype client and server applications are described. The prototype described is still a fully operational application and currently available, from the Lancaster TIC, for tourists to the city of Lancaster to use [TIC,01].

3.3.1 End System Selection

A wide range of mobile devices for use as the GUIDE end-system were considered in close consultation with the Lancaster TIC [Davies,98a]. These devices included pen-based tablet PCs, such as the Fujitsu Stylistic [Fujitsu,01] and TeamPad [Fujitsu,98], Windows CE based machines, such as the Casio Cassiopeia [Casio,99], and other PDAs such as the Apple Newton [Apple,98]. After much deliberation, we chose to use the transflective version of the Fujitsu TeamPad 7600 as the GUIDE end-system, as shown in figure 3.1.

Figure 3.1 - The GUIDE end-system

The rationale for choosing the TeamPad over the other models is presented below.

- The TeamPad has a transflective screen that enables the unit's display to be readable even in direct sunlight. Furthermore, the display is of sufficient size and resolution to present both textual and graphical information with the required lucidity. This offers a major advantage, in terms of screen real estate, over smaller displays found on PDAs. The larger display should reduce the amount of scrolling involved when reading information and should allow use without a stylus (e.g. users using their finger to navigate the UI).
- The unit is sufficiently light (850g) to hold with one hand and the inclusion of a shoulder strap allows the device to be carried easily for an extended period

of time. Furthermore, the unit is relatively resistant to rough treatment due to its ruggedised design.

- The TeamPad is based around a Pentium 166 MMX processor and therefore has sufficient power to run the GUIDE application with reasonable performance. The interactive nature of the proposed system makes the performance issue crucially important since, unlike other network-based tourist guide systems, GUIDE utilises the local processing power to provide visitors with dynamically tailored tours of the city; since the tour creation algorithm is computationally intensive, the longer the time taken to generate a tour, the greater the likelihood of a user of the system becoming frustrated. The performance of mobile devices such as the CE based devices available at the time (late 1997) were very disappointing, even for rendering web pages.
- The TeamPad is capable of running the Microsoft Windows 95 operating system and hence all of the development kits and drivers associated with this operating system were available. At the time, the Apple Newton and CE based devices did not include adequate driver support, i.e. drivers for appropriate PCMCIA based wireless networking cards.

3.3.2 Communications Infrastructure

3.3.2.1 Overview

One of the primary aims of the GUIDE system was to investigate how high-bandwidth cell-based wireless networking infrastructures could be utilised to disseminate dynamic information to handheld GUIDE units. In particular, an infrastructure was required capable of enabling GUIDE units to achieve high-speed access to general web resources. Furthermore, we also wanted to explore how the cell-based nature of the network could be used to provide GUIDE units with positioning information [Davies,01]. More specifically, the communications support for GUIDE was developed to address the following additional requirements [Davies,99]:

- **Scalability**: The system must be capable of supporting a potentially large user community requiring access to data simultaneously.

- **Flexibility of Services**: The system should support data broadcast and interactive services. Thus, it must provide a high bandwidth down link channel for the broadcast of data and include slots within the broadcast schedule to enable clients to make explicit requests for resources.
- **Support for Disconnected Operation**: Since network coverage throughout the city will not be complete and black spots will be present, the system must be able to survive periods of network disconnection and, where possible, this should not adversely disrupt the services to visitors as they explore the city.

3.3.2.2 System Architecture

The wireless network used for GUIDE is based on Lucent Technologies' 802.11 compliant ORiNOCO [ORiNOCO,01] (formerly WaveLAN) system which operates in the 2.4GHz ISM (Industrial, Scientific and Medical) band and offers a maximum bandwidth of 2 Mbps per cell. Currently, the GUIDE infrastructure consists of six communication cells deployed within a region of the city popular with tourists. A single Linux-based *cell-server* is associated with each geographic area (cell) and contains two network interfaces. The first interface provides wireless access to mobile GUIDE units within that geographic area while the second interface is used as a link back to the university campus network. Some cell-servers were installed within university owned premises around the city and so the link back was via a leased line. In buildings owned by the City Council, BT Keyline links based on the Symmetric DSL technology and providing a two-way 2 Mbit leased line were installed.

The range of ORiNOCO is approximately 450m in free space although ORiNOCO signals have very poor propagation characteristics through buildings and, therefore, by the strategic positioning of cell-servers, we have been able to create relatively small and asymmetric cells. Within the context of GUIDE this is a positive feature because by creating smaller, non-overlapping cells more accurate positioning information can be provided. For example, a communications cell for the TIC is in place which covers the immediate area surrounding the TIC but does not (by a small number of metres) crossover into the cell associated with the nearby castle. In order to manage the strategic positioning of cells, it was necessary to repeatedly place the base

station antenna in different locations to determine how the range of the cell had been affected (through the use of test software supplied by Lucent technologies).

An additional benefit of producing small non-overlapping cells is that it does not require the use of the ORiNOCO roaming facility. Had roaming been enabled then additional and unwanted network traffic would have resulted to support handover as units moved between cells. By disabling the roaming facility, GUIDE units are effectively silent on the network unless they request information that is not on the current broadcast cycle [Davies,99]. Furthermore, it has been impossible to create accurate cell boundaries due to signal reflections, interference and changes in propagation patterns resulting from fluctuations in the density and placement of objects within the cell. For example, a cell located near Lancaster's public square shows a varying degree of propagation down the square's access roads with prevailing weather conditions.

3.3.3 Communications Protocol

3.3.3.1 Overview

The overall GUIDE system, shown in figure 3.2, may be viewed as a central web server accessible by mobile clients via the wireless network. In order to improve the performance of the system, caches (cell servers) are placed in each cell and user requests are, where possible, satisfied by this local cache (geographically).

In addition, periodic broadcast schedules containing a subset of the contents of each cell's cache are transmitted to users within the cell. In this way, users who enter a cell receive (and cache locally) frequently requested pages for that geographic area. This approach has a number of benefits:

- **Improved response times**: Many of the pages which users request will already exist in their local cache (on their mobile unit) and consequently response times will be extremely rapid.
- **Improved scalability**: Since clients receive many of the pages they require without transmission the system scales well as the number of end-systems

increases. The exact extent to which the system scales depends on the pattern of user requests and this is discussed later in this section.

- **Power saving**: The ORiNOCO cards utilise less power when receiving than when transmitting. By reducing the need for clients to transmit requests there is a corresponding decrease in power consumption.

The precise benefits of the broadcast based approach depend on a number of factors. Most crucially, in a situation where there is a large degree of uniformity of page requests (i.e. most users requesting the same subset of pages such as local maps) the system scales extremely well. As the uniformity of requests diminishes so the gains over a conventional request-reply system are reduced. In the worst case (i.e. no commonality) the system will perform worse than a conventional system since time will be spent re-broadcasting pages which are not required.

While ensuring the conformity of user requests would be a significant problem in a general purpose system, within the context of GUIDE a user's access patterns will be largely dictated by their physical location. As a consequence, it is expected that each cell's cache will gradually build up a broadcast schedule containing information relating to the physical location of the cell server.

Additional factors which often affect the performance of broadcast based systems are the degree to which parameters such as the number (length) of pages broadcast, the frequency with which pages are broadcast and the strategy for replacing pages in the broadcast schedule can be tuned to match application requirements [Imielinski,94]. This may be less critical for the GUIDE system than for most other broadcast based systems. In particular, a key objective is to reduce the time a user perceives they are waiting for pages, i.e. the number of pages which have not already been cached when the user attempts to access them. If an assumption is made that each end-system has an in-memory cache of, for example, 6 MB then this is ample to cache almost all of the information relating to a given cell. Over a 2 Mbits/sec wireless network it would take approximately 30 seconds to completely fill such as cache: substantially less time than it would take most users to read the first page of information relating to a given geographic area. Furthermore, it is possible to delay presenting information to users regarding a new cell until sufficient pages are cached since a user has no means of knowing that they have entered a new cell until they are informed by the application.

Hence, assuming a user does not deviate substantially from the normal access patterns they will experience almost no delay in accessing information. The relatively high network bandwidth, large caches and predictable access patterns found in GUIDE conspire to make fine tuning of the classic broadcast strategy parameters unnecessary.

Two further responsibilities of the GUIDE cell-servers are providing position information and facilitating Internet access to GUIDE end-systems. In more detail, the cell severs periodically broadcast beacons (a UDP datagram encapsulating a location identifier) to inform GUIDE end-systems of their current cell. This information is received by the client applications' position sensor (see section 3.5.7) to notify visitors of their new location. Furthermore, cell-servers provide gateway functions to enable the GUIDE end-systems to access services on the fixed network such as accommodation reservation systems.

3.3.3.2 Engineering issues

The broadcast based caching system utilised in GUIDE uses proxies running on both mobile end-systems and cell servers. The cell server proxies build up lists of resources (i.e. HTML pages and images) to broadcast and periodically transmit these resources within their geographic area (cell). Each broadcast schedule includes an index to the contents of the schedule at the start and end of each transmission enabling mobile units to determine the contents of the broadcast cycle (see figure 3.2).

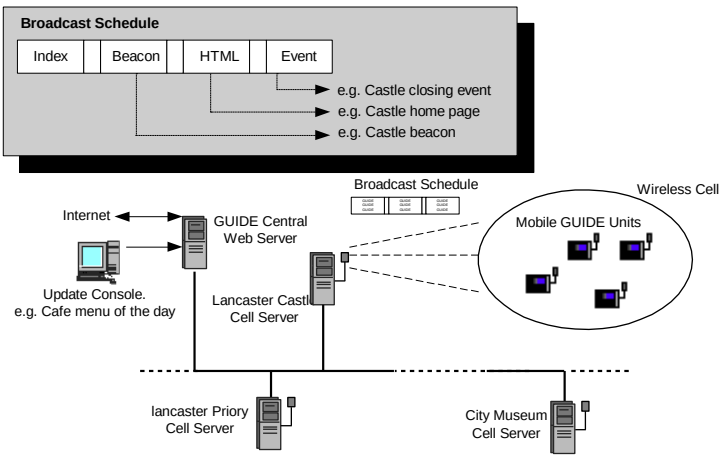

Figure 3.2 - GUIDE system architecture and broadcast schedule

All information is multicast to a well known IP multicast address to which all mobile clients are subscribed. It is possible to pre-heat the server-side proxies list of resources

to broadcast in order to reduce the amount of requests made given a cold start on the client side. Since the server side application runs on a wide range of hardware ranging from Intel 486 Processor based PCs operating at 33 Mz containing 4 MB RAM to dual processor Intel Pentium 200Mz processors containing 64 MB RAM, there are a number of adjustable broadcast parameters (shown in figure 3.3) including:

- **Broadcast delay (BD)**: The time (i.e. delay) between the end of the current broadcast schedule and the start of the next schedule.
- **Inter-resource delay (IRD)**: The time (i.e. delay) between items within the broadcast schedule to be broadcast.

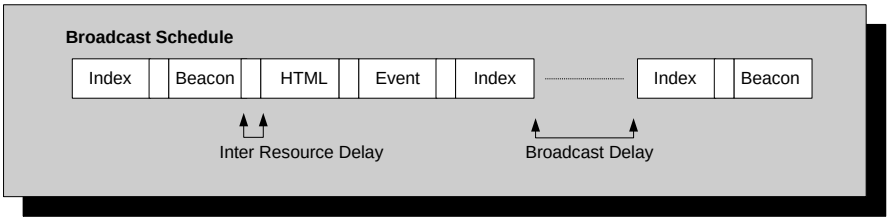

Figure 3.3 - The GUIDE broadcast schedule

The broadcast schedule is implemented as a Vector of entries in Java where each entry in the Vector may be tagged according to its type, as shown in figure 3.4. The types identify each data item as either a request, a response, an item to be cached, a service item (i.e. update) or simply an index to the broadcast schedule.

```
public class GuideBroadcaster implements Runnable
{
    static final int REQUEST = 0;    // indicates a user request
    static final int RESPONSE = 1;   // indicates a server response
    static final int CACHE = 2;      // data item to be cached
    static final int SERVICE = 3;    // service message. i.e. object update
    static final int INDEX = 4;      // indicates start/end of transmission
}
```

Figure 3.4 - The GUIDE broadcast types

Client-side proxies listen for broadcasts and fill up their caches with the contents of the latest broadcast schedule. All data items (files or objects) in the broadcast schedule are assumed to be the most recent versions available and as such immediately overwrite any existing resource with the same filename within the local cache. If a mobile client enters a cell midway through a broadcast transmission it will only partially fill its cache but will be able to determine from the trailing index those items which it has missed but which will be retransmitted in the next cycle.

All user requests for resources are routed via the client-side proxy (see section 3.5.4) which issues explicit requests to cell servers if the resource does not exist locally in the cache and the client does not believe it is scheduled for transmission soon (determined from the index). A request received by a cell-server proxy for that geographic area which will fetch the required resource if it is not already cached locally and re-transmit the response to the requesting client before scheduling the page for transmission as part of the next broadcast cycle.

Since GUIDE does not make use of Mobile IP or WaveLAN roaming, mobile clients which require only those pages contained in the current broadcast cycle need never make explicit requests of the cell servers.

3.4 Modelling Context-Sensitive Information in GUIDE

3.4.1 Overview

The GUIDE information model was required to represent and store the following distinct types of information [Cheverst,98]:

- contextual information, which can be tailored to reflect a user's context.
- geographic information, which can be expressed either in geographic (e.g. 'at co-ordinates x,y') or symbolic terms (e.g. 'in the museum') [Leonhardt,98].
- hypertext information, which includes global (i.e. Internet based) content such as the World Wide Web or GUIDE specific tourist content.
- active components, capable of storing state such as a visitor's user preferences and, in addition, performing specific actions or satisfying certain requests.

Although, individually, each of these information types has been successfully modelled, at the time of development there were no suitable models capable of handling the full complement of information types described above. For example, the current data models that have been designed for supporting context-sensitive information, e.g. stick-e-notes [Brown,96], are not well suited for managing geographic information. More specifically, such models require additional mechanisms if they are to be made capable of reasoning about the proximity of

context-sensitive nodes and answering questions such as “what locations are near me?”. Similarly, the data models supported by the current range of object oriented Geographic Information Systems are inappropriate for representing context-sensitive information [Coyle,97]. In particular, such systems lack the necessary triggering mechanisms required for handling the events raised by changes of context.

The lack of an appropriate model for satisfying the above requirements provided the motivation for developing a new information model based on the concept of integrating an active (geographic) object model with a hypertext information model. The remainder of this section describes the main components of the GUIDE information model, namely location objects, navigation point objects, neighbour relations and hypertext information as depicted in figure 3.5.

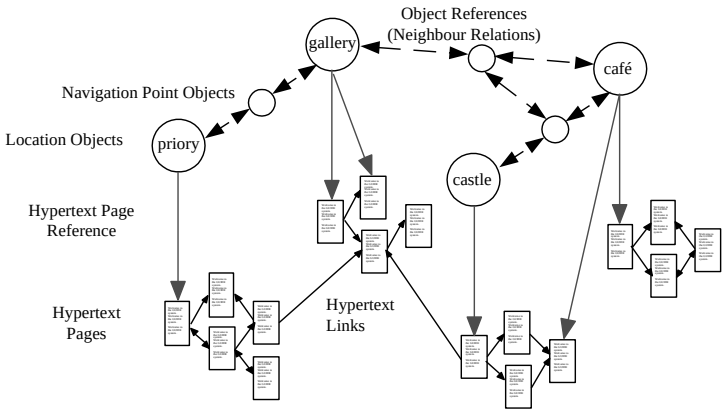

Figure 3.5 - The GUIDE information model

3.4.2 Modelling Location

3.4.2.1 Overview

The object model is used to represent entities within the city and also model the relationships between these entities. The model therefore consists of places (`GuideLocation` and `GuideNavigationPoint` objects) and relations between those places (`GuideNeighbour` objects). The inclusion of two distinct types of object to represent places (locations and navigation points) is required since a single class for representing entities would not be sufficient, for reasons described below.

3.4.2.2 Location Objects

Location objects are the fundamental building block of the model and used to represent physical locations (such as tourist landmarks) within the city. Location objects inherit the geographic properties of the `NavigationPoint` (see section 3.4.2.3) base class and include additional functionality to enable the state information to be accessed and modified.

The `GuideLocation` class is implemented as an abstract class. This means that the class cannot be instantiated directly but that specific instances, such as `NavigationPoint` can be created as sub classes of type `GuideLocation`. `GuideLocation` therefore acts as a common interface and allows standard methods to be defined. Furthermore, sub classes are able to overwrite these methods should individual location objects be required to include extra functionality. In more detail, this approach offers the following advantages:

- **flexibility**: Since specific location instances can be tailored accordingly while maintaining a standard interface.
- **extensible**: A common interface enabling access to all subclasses allows subclasses to be added/removed without changing the rest of the application.
- **reusability**: Since the common interface is declared as abstract, method declarations and their default behaviour can be provided. Therefore, only changes to the individual sub classes are required should anything other than default behaviour be required.

```
public abstract class GuideLocation extends GuideNavigationPoint implements
Serializable
{
   private GuideContext dynamicInfo; // represents contextual information
   private String photoAddress;      // URLs for location descriptions,
   private String summaryLink;       // brief summaries and images
   private String summaryText, descriptionText;
   private String commentsLinks;
   private Vector comments, nearbyPlaces;
   private int commentsTotal;
   private GuideInterests supportedInterests;
   GregorianCalendar openingTime, ;    // opening and closing times
   int minDuration, maxDuration;      // time visitors spend at attraction
   int averageDuration;
   float minCost, maxCost;           // approximate cost of attraction
   int[] timeWeights;                // weight the opening hours
   boolean[] openDays;               // days of the week attraction is open
}
```

Figure 3.6 - The GuideLocation interface

One example of a location object might be a specific museum. This museum object would include state representing its physical location within the city, opening times, the current exhibition and links to other nearby locations. The `GuideLocation` class includes access methods which allow the context associated with that location to be accessed, queried and updated by other components within the system. The `GuideLocation` interface contains the following methods:

```
public int evaluate(GregorianCalendar time, GuideContext thisContext)
```

This method is used by the system's tour creation component (see section 3.5.10) and performs an evaluation of the suitability of visiting that particular location based on some contextual information. For example, a call to `castle.evaluate(T,C);` would cause the object relating to the castle to be evaluated based on an arrival time T given Context C. This context could include information relating to the visitor profile, their budget allowance and the current weather forecast. The integer value returned is used to suggest a suitable tour to the user. The algorithm is described in more detail in section 3.5.10.

```
public String get(String name)
```

This generic interface allows access to the object's variables and associated context. For example, a call to `tic.get("NEIGHBOURS");` would cause the object representing the tourist information centre to return a list of object references for those landmarks that are immediate neighbours. Using this list of object references the GUIDE system is able to carry out route-finding and navigation tasks as well as placing context-sensitive nodes in a geographic context, for example, helping answer user queries such as "What attractions are in this area?".

```
public void execute(String command)
```

The execute method allows specific method calls to be executed by passing as a parameter the name of the method to call. For example, when a user enters a historic landmark such as Lancaster Castle the following method `castle.execute("PlayWelcome")` could be invoked to play an audio or video stream.

3.4.2.3 Navigation Objects

Navigation point objects are used only to support the construction of guided tours and for route guidance purposes and represent way-points between location objects. Navigation point objects contain purely geographic state (as shown in figure 3.7) that can represent a point within the city (e.g. a road junction or cross roads).

```
public class GuideNavigationPoint implements Serializable
{
   private float longitude, latitude;// GPS Coordinates
   private String name, fullName;    // e.g. castle, Lancaster Castle
   private String inRegion;          // unique id of cell server
   private int id;                   // System wide unique id
}
```

Figure 3.7 - The GuideNavigationPoint interface

3.4.2.4 Modelling A City

Relationships between objects within the geographic model are represented in two distinct ways. First, location objects may contain *references* to other location objects within the model, or, *relationships* can be created between location objects as separate `GuideNeighbour` objects. The first type of relationship, called the *nearby* relationship, is used by the application to help build (dynamically) a hypertext page of information that is able to tell a user what attractions are physically nearby their current location. The `GuideNeighbour` relationship offers a more flexible and extensible approach and includes attributes (weights), modelling characteristics such as the distance or cost of travel between two points using variety of means of transport. Using this representation other components such as the tour guide object can traverse objects and relationships to determine, for example, neighbouring locations and optimal routes between any number of points. Figure 3.8 shows an example relationship between two location objects. This example shows two alternative routes between the castle and TIC (and vice-versa), each with different route guidance instructions and associated weightings.

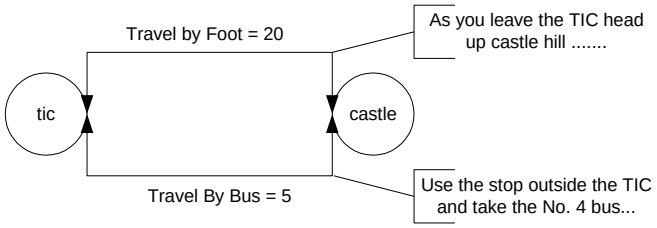

Figure 3.8 - The GuideNeighbour relationship

3.4.2.5 Modelling Context

In addition to the use of locations and navigation points as described above, each object within the model has an associated `GuideContext` object. In essence this is used to represent and maintain dynamic information, for example, context relating to its current status, that is, whether an attraction is currently open or closed. The context object, shown in figure 3.9, is modelled as a hashtable of name-value pairs. A time stamp and an expiry field is associated with any contextual change. This enables other components to query an object and retrieve its state and determine its validity.

```
public class GuideContext implements Serializable
{
    private Hashtable context;           // a hashtable of name-value pairs
    private GregorianCalendar timestamp; // time context was last updated
    private int timeout;                 // context expires (lease time)
}
```

Figure 3.9 - The GuideContext interface

3.4.3 Hypertext Information and GUIDETAGS

3.4.3.1 Overview

In addition to the object model described above a standard hypertext model to represent information in the GUIDE system has also been adopted. This allows information to be created and structured using well understood techniques.

The combination of the object model with a hypertext information repository provides an information model which satisfies the requirements discussed at the start of this section. The key feature of this approach is that it supports a high degree of coupling between the object model and the hypertext information. In more detail, the object model provides a convenient mechanism for accessing hypertext pages in much the same way as a search engine enables access to web pages. More specifically, objects

contain references (URLs) to hypertext pages providing users and application components with multiple entry points into the hypertext information base.

Crucially, hypertext pages are able to reference the object model which enables an author to construct a hypertext page which can interrogate the state of objects within the model (e.g. day, time, weather) and hence the current context of the environment. The page can thus be displayed differently depending on factors such as the number of times a visitor has been to a location or the current status of the attraction [Cheverst,01d].

3.4.3.2 GUIDETAGS

In order to allow hypertext pages to reference the object model the authors of standard hypertext pages are able to augment their pages with tags [Cheverst,98], [Cheverst,01a]. These tags may make method invocations of the object model, query the state of the model or control the display of the information to a visitor (see example in figure 3.10). These GUIDETAGS take the form of special instructions which are able to query the GUIDE object model, for example a tag might be used to represent the time of day. Thus, if the time of day was morning then the café menu would be displayed differently to that shown if the time of day was afternoon.

```
<P> Welcome to <GUIDETAG INSERT POSITION>
From here you can visit the neighbouring locations of:
<GUIDETAG INSERT NEIGBOURS>
<GUIDETAG INTEREST ((HISTORY GREATER 50) AND (ARCHITECTURE GREATER 50))
The following features will be of particular interest to you ……. <P>
</GUIDETAG>
```

Figure 3.10 - Example use of GUIDETAGs

Appendix B details the syntax and semantics relating to the range of tags in use by the GUIDE prototype application and also describes how these tags are used to create adaptive hypertext information based on both the user's personal context and the environmental context.

The information model (object model and hypertext information base) supports the tailoring of information presented to users without requiring the use of a specialised database to generate the information on a per-request basis. In addition, the GUIDE

system can operate using cached (generic) information without requiring continual access to an information service.

3.4.3.3 Generating Dynamic URLs and Content

To support the dynamic nature of the content required by the GUIDE system, the information model supports the notion of hypertext pages containing partially incomplete URLs to other resources within the information base. The system is able to inspect and dynamically rebuild URLs based on a user's context in order to retrieve the required resource. To successfully rebuild a URL, a user's preferred reading language and current location is required, which is obtained from the user profile.

In more detail, URLs within GUIDE take the form specified by Berners-Lee *et al* [Berners-Lee,94], for example, http://www.bbc.co.uk/index.html. The client agent (see section 3.5.3) inspects each URL and determines whether or not the hostname is guide specific, i.e. http://guide.lancs.ac.uk/. URLs representing hypertext pages or resources (e.g. images) stored on the web are retrieved by the proxy object normally. URLs that relate to GUIDE specific tourist information require further inspection. A complete GUIDE URL takes the form http://guideserver/language/object/resource for example http://guide.lancs.ac.uk/english/castle/history.html. A request for this resource would indicate that the resulting page would be a history related hypertext page for the castle area of the city presented in English.

The required reading language, object of interest or resource may be omitted from the requested URL and in these cases the URL is inspected and rebuilt to create a valid URL. For example, omitting the reading language causes the user profile to be queried and the user's preferred reading language to be determined and inserted dynamically. Similarly, omitting the location object required causes the system to default to using the user's current location, which is inserted dynamically.

This mechanism offers the advantage that generic pages can be created with links to other regions within the city (for example, the castle or the city museum) and the hypertext links within those pages need not include the user's reading language. This permits two users of the system with different preferred reading languages (e.g.

English and German) to follow the same hypertext link and be taken to the page they desire based on their preferred language, i.e. different pages.

3.5 The GUIDE Application Design and Implementation

3.5.1 Overview

The GUIDE prototype application was designed as a number of modules to aid application evolution and maintenance. The complete set of computational objects that make up each instance of the application prototype are illustrated in figure 3.11.

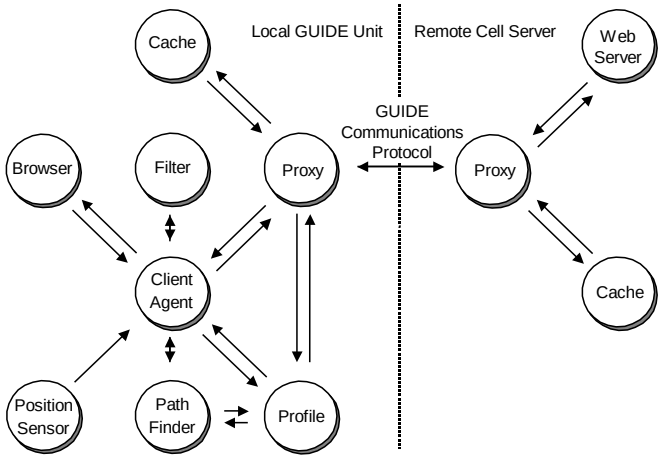

Figure 3.11 - The GUIDE application architecture

The following sections consider the design and implementation of each functional module in turn.

3.5.2 The GUIDE Browser

The GUIDE browser component represents the main user interface to the GUIDE system. In essence the browser is implemented as a Java Swing application containing an embedded web browser, Sun's HotJava HTML component [Sun,99a]. All user interactions are carried out using the web browser interface and, in addition to supporting standard navigation buttons found on most common web browsers (such as back, forward and reload), the GUIDE browser contains widgets relating to specific functionality provide by the GUIDE system. The extra functionality afforded by these

widgets includes the tour generation system and interactive services. Further information relating to the GUIDE user interface can be found in section 3.6.

3.5.3 The GUIDE Client Agent

The client agent is the name given to the application's session manager on the mobile GUIDE unit which is responsible for co-ordinating all client side object interactions. The client agent has the responsibility of starting other component threads of execution and maintains a handle to each executing thread. This allows all components within the system to interact with one another via the client agent. The client agent is also responsible for recording a user's current and previous locations within the city and for maintaining a log of all events processed by the application. These events include user interactions (such as button presses), all requests and responses for hypertext resources and other events such as hearing a location beacon from a cell server. All events are stored locally in a log file which enables an analysis of the performance of the system to be carried out at a later date. The log files were one of the mechanisms used to help evaluate the GUIDE system during the user field trial described in [Cheverst,00a]. A sample log file is shown in figure 3.12.

User location	Time	Details
tic	9:19	Last location update received 30 secs ago.
tic	9:23	Currently receiving location updates
tic	9:34	Saving User Preferences
tic	9:34	Local::http://guide.lancs.ac.uk/information/welcome.htm
tic	9:34	Local::http://guide.lancs.ac.uk/common/images/dude.gif
tic	9:34	Last location update received 30 secs ago.
castle	9:36	Currently receiving location updates
castle	9:38	Tour Button Pressed
castle	9:38	Displaying Tour Guide To User

Figure 3.12 - Sample user trace

3.5.4 The GUIDE Proxy

The proxy component accepts all incoming HTTP requests for information from the browser and is responsible for processing these requests on the browsers' behalf. The proxy first inspects the URL to determine whether or not a GUIDE specific resource or a more general web resource is required. If the request is GUIDE specific then the proxy inspects the URL further to determine its validity (see section 3.4.3.3 Generating Dynamic URLs). The proxy attempts to satisfy all requests locally from

the cache before making explicit requests of the remote cell server. Having retrieved a resource either locally from the cache or from the remote cell server the proxy then determines whether or not the page requires further processing before being handed back to the browser. If a GUIDE specific hypertext page is requested then it may contain GUIDETAGs and therefore requires processing by the filter (`GuideFilter`) component. An example call to the filter component is shown below:

```
filter.filter(loc, resource, tempLocation, userProfile);
```

This invokes the filter method contained within the `GuideFilter` component and passes as parameters the user's current location, the resource (i.e.hypertext page) to be filtered, a temporary file location to store the filtered resource (thus preserving the original copy in the cache) and a handle to the user's profile. Figure 3.13 summarises the actions of the proxy component.

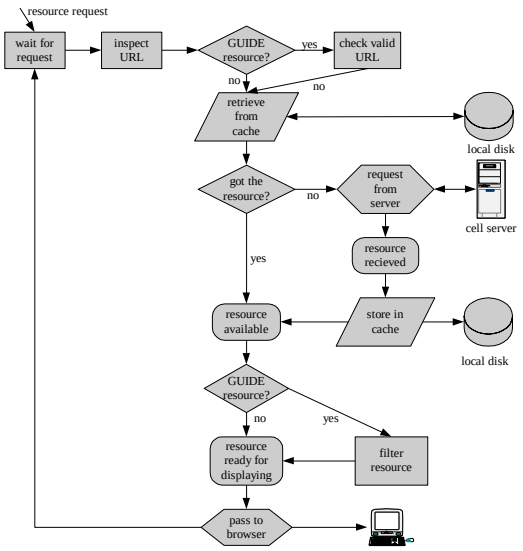

Figure 3.13 - The GuideProxy

3.5.5 The GUIDE Filter

The filter is responsible for processing GUIDE specific hypertext pages and creating dynamic (context-sensitive) content [Cheverst,00c]. The filter parses GUIDE specific hypertext pages searching for GUIDETAGs to process. The actions taken by the filter depend upon the type of GUIDETAG found. Although these were described in detail in section 3.4.3.2, to summarise the filter is able to:

- **Create Content Dynamically**: Hypertext content can be created dynamically by querying the object model to retrieve context-sensitive information. For example, when personalising a page the user profile is queried to obtain the user's full name.
- **Modify Existing Content**: Hypertext content can be shown, hidden or modified from a user depending on their user profile. For example, a hypertext page for a particular café may include the day's menu. A vegetarian user making a request for the menu results in the page being modified so that vegetarian information is located near the top of the page. Similarly, a user who has not expressed an interest in the city's many historical landmarks will receive hypertext pages without the historical content.

3.5.6 The GUIDE Position Sensor

The position sensor is responsible for notifying system components of changes in a user's physical location within the city. The position sensor joins and listens on a well known multicast address for location updates represented by beacons from cell servers. To avoid problems at cell boundaries where a user may receive location beacons from multiple sources, a smoothing function is used that dictates that the user must 'hear' at least N consecutive beacons from a new cell server before notifying this new location to system components. On hearing a new beacon, for example the beacon relating to Lancaster Castle, the position sensor notifies the client agent of this change in location. The client agent then instantiates the object relating to this new beacon, in this example the object representing Lancaster Castle. All locations are represented by the `GuidePosition` component (shown in figure 3.14) which contains the location object relating to the users current location and of all the nearby locations. Since a user is likely to request information relating to either their current location or attractions located close by, this improves the response times should a GUIDETAG be processed that requires the object model to be queried. Finally, the client agent notifies the browser of the new location and the user interface is updated accordingly (see figure 3.17 showing a screenshot of the application in use).

```
public class GuidePosition
{
    // Store a user's current and previous location
    private GuideLocation currentLocation;
    private GuideLocation previousLocation;
    private GuideRegion currentRegion; // user's current region
    // Maintain a table to hold nearby locations objects
    private Hashtable nearbyPlaces;
    // Maintain a handle to the client agent component
    GuideClientAgent handle;
}
```

Figure 3.14 - The GuidePosition interface

3.5.7 The GUIDE Cache

The cache is responsible for maintaining a local repository of hypertext pages, related resources and location objects for retrieval by other system components. This avoids the need to make explicit remote requests for resources and increases the response times perceived by a user. The cache component joins and listens on a well known multicast address for cache updates broadcast by the cell server as part of the broadcast schedule (see section 3.3.3.2). The cache replacement policy is based on the assumption that data items received as part of the broadcast are the most recent versions available and should therefore replace any previous items.

3.5.8 The GUIDE User Profile

The user profile component is responsible for storing a user's personal details and is queried by other GUIDE components to enable context-sensitive content to be displayed to a user. These are decided by the user on application initialisation and do not generally change during the lifetime of a session. The user profile contains a number of attributes including:

- **Personal attributes**, such as username, group name (if touring as part of a group), full name, reading language and age.
- **Tourist interests**, represented by the `GuideInterests` class, represents user interests as a hash table of name and value pairs. For example it may contain entries such as "Vegetarian = 0; History = 100" which indicates that the user is interested in history but is not a vegetarian.

- **Activity**, context relating to a user's current activity, such as whether they are currently busy or free. This information can be used by others to determine when to interrupt someone via the message service, described in section 3.7.5.

Additionally, the user profile maintains a history of all the attractions visited and actions carried out by a user. In more detail, the attractions visited, the tours followed and the resources requested are all recorded and stored as part of the profile and used to personalise hypertext pages when a user enters an area of the city (for example, "Welcome to…." or "Welcome back to …"). This information could also be used to create a personalised brochure or hypertext document as a souvenir of their visit. The `GuideUserProfile` interface is shown in figure 3.15.

```
public class GuideUserProfile implements Serializable
{
    // constants to represent user activity
    public static final int FREE = 0;
    public static final int BUSY = -1;

    // generate a unique id based on the username, date and time
    private String SessionName;
    private String userName;          // e.g. Keith
    private String password;
    private String fullName;          // e.g. Keith Mitchell
    private String groupName;         // e.g. School or family name
    private String language;          // e.g. English, French or German
    private GuideInterests interests; // e.g. History, Music, etc.

    private boolean contactable;   // user wishes to be contactable?
    private boolean anonymous;     // user wishes to remain anonymous?

    private int currentContext;    // Store current context, e.g. on a tour

    private Vector toursFollowed;  // Store any tours followed
    private Hashtable visited;     // Store where a user has visited
    private Hashtable visitedPages;// Store resources accessed and frequency
}
```

Figure 3.15 - The GuideUserProfile interface

3.5.9 The GUIDE Notification Dispatcher

The notification dispatcher is responsible for receiving events from the remote cell servers and processing them appropriately. The dispatcher joins and listens on a well known multicast address for incoming notifications (or events) and, dependant upon the type received, is able to carry out the following actions:

- **Update**: Notifications may contain object model updates which could either be a change in context relating to a particular attraction or a new object to

replaces an existing object. For example, an attraction closing early would require a notification to be dispatched to notify the mobile GUIDE units that this attraction is no longer available. Similarly should a landmark change (for example, a shop being sold and converted into a café) then this information also needs to be updated on the mobile GUIDE units.

- **Display**: Notifications may contain data that needs displaying to the user. These take the form of messages and allow users of the GUIDE system to communicate with one another. In addition, the tourist office is able to dispatch events that they feel may be useful to users of the system, for example, warning users of traffic congestion on the city one way system.
- **Execute**: Notifications may also be used to invoke methods on the mobile client. For example, the tourist office may wish to collect the log files stored on the mobile clients and therefore dispatch an event causing the mobile clients to return their log files to the tourist office.

Notifications are represented within GUIDE using the `GuideNotification` class (shown in figure 3.16) and may be directed towards a number of different components within the information model, such as:

- **Location objects**: Notifications intended for location objects are used to update state pertaining to that object or to invoke the methods of that object.
- **User profile**: Notifications intended for the user profile object are context-sensitive notifications and may not be intended for all users. Therefore, the attributes stored in the incoming event are compared to the user preferences stored in the profile to determine whether or not the event was intended for this recipient. For example, the tourist office may dispatch an event relevant to vegetarian users or users with an interest in history.
- **Users**: Notifications taking the form of messages to be displayed may also be context-sensitive. In more detail, messages may be intended for a specific user, a group of users or all users of the GUIDE system. Support for sending group messages enables family members or groups of school pupils using the system to communicate with the rest of their party using a group identifier.

Finally, to enable events to be delivered in an appropriate and timely manner context-sensitive attributes (delivery types) are supported to control the delivery of notifications. First, notifications may be delivered if certain contextual conditions are satisfied. For example, a notification relating to a musical event such as a concert may be tagged with a musical contextual attribute. Once received, the notification will be compared against a user's preferences and will be delivered if they are interested in this topic. Second, notifications may be delivered if the intended receiving object is the user's current location, within a specific geographic area or if the object is currently active. Currently active refers to an object representing the user's current location or its immediate neighbours. This allows notifications to be processed at particular points along a guided tour. Finally, notifications may be delivered instantly.

```
public class GuideNotification
{
    // Notification types supported
    public static final int UPDATE = 0;     // Update the object model
    public static final int SHOW = 1;       // Display events to users
    public static final int EXECUTE = 2;    // Execute some code
    // Delivery types supported
    public static final int ALWAYS = 0;
    public static final int IFCURRENT = 1;
    public static final int IFINREGION = 2;
    public static final int IFACTIVE = 3;
    public static final int IFCONDITION = 4;

    // Recipient types supported
    public static final int OBJECT = 0;     // an object update
    public static final int REGION = 1;     // objects within a region
    public static final int INTEREST = 2;   // user interests
    public static final int USER = 3;       // a user (e.g. message)
    public static final int USERGROUP = 4;  // a group message
    public static final int ALLUSERS = 5;   // a message to all
    public static final int SERVERACK = 6;  // Server Acknowledgement
    public static final int USERACK = 7;    // User Acknowledgement
}
```

Figure 3.16 - The GuideNotification interface

3.5.10 The GUIDE Path Finder

The path finder or tour guide component is responsible for traversing the location and navigation point objects and creating a representation of the city of Lancaster. This model of the city can then be used to generate personalised tours of the city.

The creation of a tour of the city does not simply aim to create the shortest route between attractions within the city, but offers a more intelligent tour based on a

variety of factors. The tour creation component was designed after consulting a professional human guide who would take into account issues such as [Davies,01]:

- **Visitor interests**: Any guide should offer a tour which reflects the interests of the visitor.
- **City attractions**: Visitors often want to visit a city's main attractions even if these do not fall directly within their normal interests.
- **Travel Constraints**: Many visitors have mobility constraints, either physical or financial, that restrict a visitors travel plans through the city should be catered for (for example, wheelchair users).
- **Available time**: Most visitors have a limited time to spend in a city and tours have to be created to maximize time without making the visitor feel rushed.
- **Time sensitivity of attractions**: In many cases there are good and bad times to visit specific attractions. For example, a museum that is open to parties of school children in the mornings may be best avoided if a visitor is hoping for a quiet tour of the museum. Similarly, on a day with showers forecast for the morning and sun in the afternoon then it may be desirable to visit a park in the afternoon during the warmer hours of a day and take an indoor tour first.
- **Weather**: For many outdoor attractions the prevailing weather conditions make a big difference to the amount of time visitors spend at a particular attraction. The weather also affects the desirability of certain travel paths. For example, having a tour-guide that suggests a stroll along a river bank followed by a picnic when it is raining will clearly be ignored by users.
- **Budget constraints**: Many visitors have financial constraints and a tour-guide needs to be sensitive to these issues.

The calculation of a tour therefore depends on a variety of factors and through achieving the correct balance between different demands on the visitor's time and money. Furthermore, since many of the factors described above may vary over time it is often necessary to recalculate tours during the course of a single day. For example, a user may spend longer than anticipated at a particular attraction which means the user has less time to spend at other attractions in the city.

To meet the above requirements, when constructing a guided tour of the city, the path finder attempts to quantify or evaluate the quality of a tour by allocating numeric values to attractions and routes between attractions. These values are influenced by both the current context (weather, time etc.) and the user's preferences and tours can be ordered by comparing their total scores. In more detail, the system dynamically generates a full set of tours that encompass the attractions requested, produces a total score for each permutation and then recommends to the visitor the tour with the highest score (i.e. most suitable).

In addition, the system incorporates a number of "tricks" to try and improve a given tour's overall score. For example, consider the following scenario. It is 9.30am and a tourist has just picked up a GUIDE unit from the TIC. They ask the system to generate a day-long tour that takes in just two places, a nearby museum (which opens at 10.30 am) and a park on the other side of town. The system recognizes that the best order is to visit the museum first and then spend the afternoon in the park and will suggest "padding" activities (e.g. nearby cafés, exhibitions or related landmarks) to occupy the visitor until the museum opens. The tour-guide generation system uses the geographic (object) model described in section 3.4, with each location object storing a range of scores reflecting the appropriateness of visiting that location for a particular set of context inputs. A detailed flowchart and pseudo-code algorithm describing the tour creation process is shown appendix C. It is worth noting that the solution described above offers a relatively simple pragmatic solution to the problem of dynamic context based route finding. Although the solution works well for the relatively small amount of context modeled by the GUIDE system, it is clear that the problem could obviously scale up to a large AI problem if based on accurate models of the environment.

3.6 The GUIDE User Interface

The GUIDE user interface (as illustrated in figure 3.17) is based around a modified browser metaphor. This decision was made for two reasons. First, the metaphor closely matches the kind of information modelled in GUIDE, i.e. the notion of following hypertext links in order to access greater levels of detailed information about a particular attraction or event within the city.

Second, the browser metaphor was based on the growing acceptance of the web and the increasing familiarity of the metaphor as a tool for interaction. It was hoped that positive transfer from the use of common web browsers would help make the system both easy to use and easy to learn for users with previous web experience. However, we also wanted to ascertain the extent to which the basic metaphor would be appropriate for the task of supporting the additional context-aware based functionality required by GUIDE. More specifically, we wanted to investigate the extent to which differences and inconsistencies with the standard would prove confusing to users.

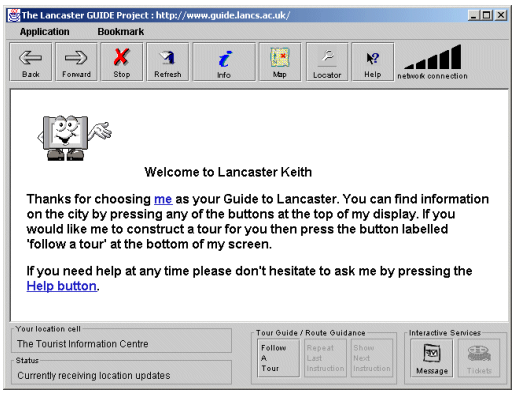

Figure 3.17 - The GUIDE user interface

In order to help the system appear more approachable to visitors the idea of using a *buddy* was adopted to give the GUIDE system a friendly personality. The initial aim was to extend this feature to include a number of themed characters, similar to those found in Microsoft Office, for example a medieval tour guide or a Victorian tour guide. This decision was based on the observation that, in general, novice users will find a computer-based interactive system more approachable if it is perceived as having a polite and friendly personality [Oberlander,97]. However, as users become more expert with using the system then this approach may become less appropriate.

Other user interface design decisions included considering the use of a multimodal user interface, e.g. one in which information and navigation instructions could be delivered to the user using speech output [Cheverst,01b]. However, following discussions with staff at the city's TIC, this approach was not pursued. The main reservation was based on the fact that GUIDE is designed for outdoor pedestrian use and concerns were raised relating to visitors being distracted by the system when crossing roads if the system chose that time to deliver information. Following on from

this point, in general we have reservations about the effective bandwidth of voice for information transfer and the extent to which users can control the pace of information delivery with a speech-based system.

3.7 Application Functionality: Some Scenarios

3.7.1 Overview

The GUIDE system provides city visitors with a wide range of functionality: a visitor can use their GUIDE unit to access context-aware information, create tailored tours of the city, access interactive services, and send and receive text messages.

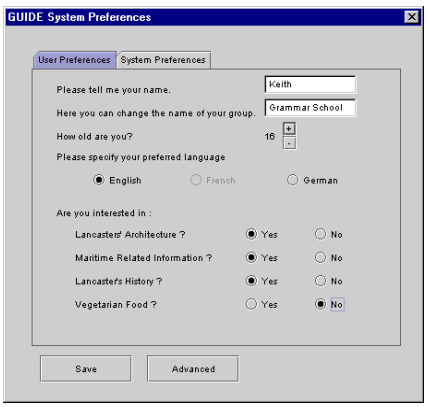

Figure 3.18 - Choosing visitor preferences

When the application is first initialised, a new representation of the city of Lancaster is created. Following this, the user is prompted to enter their visitor preferences (using a wizard similar to that shown in figure 3.18). Once the user has specified their interests the system welcomes them to the city of Lancaster (see figure 3.17).

3.7.2 Accessing Context-Aware Information

A visitor can use GUIDE to retrieve web-based information based on their current context, e.g. location and user preferences. In addition, the system also enables visitors to access web pages in a non context-aware manner. More specifically, by following hypertext links visitors have general access to the World Wide Web.

In order to retrieve information the visitor can tap the 'Info' button and this will cause a set of choices to be presented in the form of hypertext links. Figure 3.19 shows the

choices that would be available if the visitor happened to be located in close proximity to the tourist information centre (TIC).

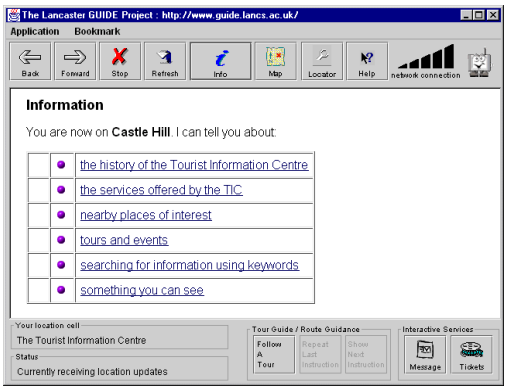

Figure 3.19 - Accessing information using GUIDE

It is important to note that not all the options are based on the visitor's current location. For example, the visitor is provided with the ability to search for information using a keyword search facility, irrespective of their current location. Early versions of the system did not offer the search facility and in effect restricted the scope of information available to a visitor to that closely related to the visitor's location. During an expert walkthrough of the system it became apparent that constraining the visitor's access to information based on their location can be very frustrating for the visitor if the information they require cannot be accessed because we did not deem it to be of sufficient relevance to the area concerned [Cheverst,00b].

On a more general point, our experience with this aspect of the system has taught us that designers of context-aware systems must not be over zealous when deciding to constrain the information or functionality provided by the system based on the current context. The difficulty of pre-empting the user's goal is further highlighted when considering the situation where the visitor selects the third option in order to view a list of nearby places of interest. When presented to the visitor, the list is sorted in such a way that those attractions that are open, and have not already been visited, are placed higher up the list. The assumption is made that the visitor is more likely to be interested in attractions that are open and that have not already been visited. An earlier version of the system constrained the output by removing all closed attractions from the presented list. However, this frustrated some visitors who were interested in

visiting the attraction anyway, e.g. to view the architecture of a building [Cheverst,00b].

3.7.3 Create a Tailored Tour of the City

The GUIDE system is designed to enable visitors to request a structured tour of Lancaster based on a set of attractions that they wish to visit. In order to ascertain this set of attractions, visitors are asked to choose any number of pre-defined tours of the city or select attractions from a set of categories such as 'Historic' and 'Recreation'. The first option was added after an initial evaluation with a small number of visitors to the city. It was found problematic to ask visitors themselves to choose attractions to be included in their tour since they did not necessarily appreciate what was special about the city of Lancaster or have an appreciation for the relative distances between attractions. For example, everyone would like to drink stout when in Dublin even though they might not do so at home. For this reason, we included pre-defined tours and a 'Popular Attractions' category to aid visitors in their choice of attractions to visit.

Once a tour has been successfully generated (see appendix C and [Davies,01]) for more information), the visitor is presented with a recommended sequence for visiting their chosen attractions. The visitor can then agree to be taken to the next attraction in the suggested sequence or override this recommendation by selecting a different attraction to be their next destination. A structured tour is broken down into a series of distinct stages. Once completing each stage, users may request the system to describe the next stage of the tour by touching the 'show next instruction' button. Stages in the tour are described using a navigation instruction. These instructions provide the visitor with a piece of text explaining in some detail how to get from their current location to the next attraction in the tour, as shown in figure 3.20.

Figure 3.20 - The presentation of navigation information

It is important to note that, given the same set of attractions, the ordering of the tour recommended by the system can actually change dynamically. This can occur when, for example, a visitor stays at a location longer than anticipated or if one of the attractions announces that it will close early. The system regularly calculates whether or not the current order for visiting the remaining attractions is appropriate given current time constraints.

3.7.4 Access Interactive Services

By providing remote access to interactive services, such as the booking of hotel accommodation, visitors can save time by making bookings via their GUIDE unit. In addition to providing remote access to services provided by the TIC, the GUIDE system also enables access to other services, such as enabling visitors to query films on show at the local in addition to enabling visitors to reserve seats remotely.

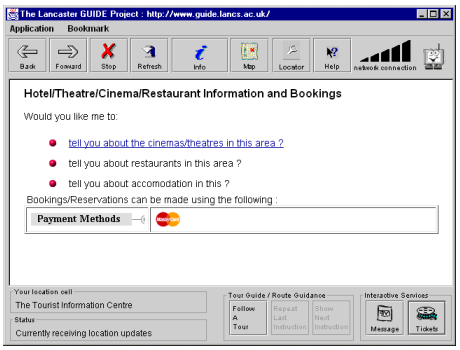

Figure 3.21 - GUIDE interactive services

3.7.5 Send and Receive Messages

The messaging service enables groups of visitors, who may have separated in order to visit different attractions in the city, to keep in touch and also enables visitors to request information from staff at the TIC. Similarly this service allows members of the TIC to send out announcements to all users of the system, as shown in figure 3.22.

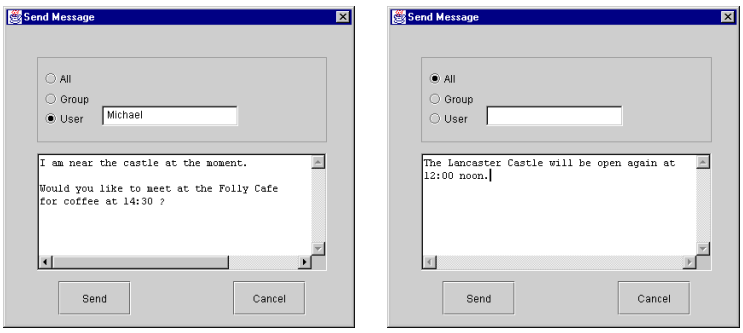

Figure 3.22 - GUIDE messaging service

3.8 Conclusion

This chapter has focussed on the work carried out during the GUIDE project and has detailed the design and implementation of a prototype context-aware application built to satisfy the requirements of tourists to the city of Lancaster. In addition, this chapter has also provided a number of insights relating to the challenges involved in deploying a city-wide wireless network and a prototype context-aware application designed for use in a mobile environment. The GUIDE prototype has acted as the author's main research vehicle for investigating the area of mobile context-aware computing and chapter five analyses the GUIDE approach more closely to identify a number of shortcomings regarding the approach presented in this chapter. This analysis is used to inform a set of requirements which in turn act as the basis for the more general service based approach to context-aware application design described in later chapters.

Chapter 4

Requirements and Design For A Context Service Based Architecture

4.1 Introduction

This chapter describes the design of a context service designed to provide support for building mobile context-aware applications. The context service demonstrates a flexible and dynamic approach to supporting both application development and operation within mobile environments. More precisely, the context service is designed to provide the application programmer with a convenient mechanism for supporting mobility and application extensibility through an appropriate layer of abstraction between applications and infrastructure.

The chapter commences by presenting a general set of requirements used to influence the design and implementation of the context service based architecture. The requirements are derived through an analysis of the related work detailed in chapter two and by means of a reflective critique of the GUIDE implementation presented in chapter three. This analysis highlights the primary limitations present within the research field in general and the GUIDE prototype when considering mobile environments. Following this, the overall architectural features and computational

objects relating to the context service are described before chapter five goes further to consider the context service from an engineering perspective in order to detail the implementation and protocols incorporated into the context service. Chapter five concludes by describing a number of context-aware applications designed using the architecture.

4.2 Analysis and Requirements

4.2.1 Overview

This section uses observations obtained from chapter two in addition to an analysis of the GUIDE prototype, as presented in chapter three, to establish a generic set of requirements for supporting context-aware applications within mobile environments. The primary objective of this analysis is to identify the core architectural features required by a context service to better facilitate context-awareness within dynamic environments. The analysis is also motivated by the results obtained through a user field trial evaluation of the GUIDE prototype [Cheverst,00a] and extensions to the application functionality [Cheverst,00d].

4.2.2 Requirements for a Context Service

4.2.2.1 Overview

Chapter two focussed on two main areas of research, the impact of mobility on application design and current approaches to developing context-aware applications. The author argues that application developers must be aware of the issues relating to both of these areas of research in order to successfully author next generation adaptive mobile applications. The reason for this is twofold. First, the requirement for supporting mobile users coupled with the increasing range of available context sensing technologies establishes the need to support location based services in addition to user mobility. Indeed, current telecommunications operators are investing heavily in their next generation networks to allow access to location (context) based services from a variety of mobile handsets [BT,01]. Similarly, work in the area of context-aware computing will increasingly focus upon user mobility as a valuable form of context since to a large extent it is possible to make a significant number of

assumptions pertaining to user activity based purely on location. Therefore, the amalgamation of issues which pertain to both fields of research will be vital to the support of context-aware applications aimed at mobile environments. The requirements will be presented within the broad categories of systems support for mobile and context-aware systems.

4.2.2.2 Requirements for Mobile and Distributed Systems Support

R1: Supporting User and Device Mobility

Chapter two described how research has generally focussed on two primary types of mobility. The first involves both user and computing device mobility, for example, a field engineer using a pen-tablet and working on the move [Friday,99]. The second is more in accordance with the ubiquitous computing vision [Weiser,93] involving a mobile user utilising a richly equipped fixed networked environment (i.e. only the user is mobile [Harter,99]). Within a heterogeneous processing environment both these forms of mobility may coexist since users are likely to move between portable and fixed devices during the course of a day.

Furthermore, support for mobility must be considered at a number of different levels including the systems and the user interface level. Applications must be able to withstand periods of disconnection and perhaps offer levels of service during periods of disconnected operation [Cheverst,99a]; however, feedback pertaining to the state of the environment must also be supported as described further in requirement R13.

R2: Support Persistence of Application and User State

The GUIDE prototype described in chapter three does not adequately support users within a heterogeneous processing environment. More specifically, users are unable to initiate an application session using one device and continue that session using an alternative device. Whilst the GUIDE client application maintains some state relating to a range of environmental contexts including the user profile and the city information model, this information is only stored locally on the mobile device and is not made persistent (across sessions or devices) at run time or once the application has terminated. This acts as a major shortcoming for the GUIDE prototype since the

battery capacity of the Fujitsu TeamPad is such that it is only able to operate for up to (at most) two and half hours when powering the wireless network card before the batteries need replacing. The inability to make state persistent and retrievable whilst batteries or devices are changed presents a limitation to the system since valuable state may be lost. The inclusion of this functionality may enable a user (over a number of application sessions) to construct a detailed user profile of their activities and better facilitate sharing of personal context between sessions or other applications.

This highlights a strong requirement for providing system support for context-aware applications capable of operating in a heterogeneous processing and networking environment. Furthermore, applications should be capable of operating over different hardware and networking infrastructures despite the varying levels of service that these infrastructures afford.

R3: Support Flexible Interaction Models

Within a mobile distributed environment there exists a requirement to support a range of interaction styles ranging from highly synchronous to asynchronous, since it is not feasible to provide a one-size-fits-all approach to interaction between system components. Moreover, since interaction and performance often fluctuate as users experience varying levels of network connectivity, there is a need to consider the balance between client, server and network loads since solutions must often be able to accommodate low bandwidth wireless links, partially connected hosts, and variable communication speeds and latencies.

When considering partially connected mobile hosts, information may often be out of date. Therefore applications that make explicit requests for context changes trade off the overhead of polling versus the timeliness of the information they receive [Schilit,95]. More specifically, reducing the interval between successive polls lowers the time taken for an application to see a context change at a server, although one drawback of polling is that it may generate unnecessary network traffic when information changes infrequently.

These issues were, to some extent, overcome by the broadcast approach described in section 3.3.3, which is able to accommodate a large user community per wireless cell

[Cheverst,01d]. However, the use of broadcast increases the load in client-side processing since all incoming communications requires filtering in order to establish relevance to the receiving mobile host. An approach based on the publish-subscribe paradigm limits the extraneous network traffic created through polling through the use of callbacks from servers to clients. However, maintaining the callbacks may add complexity to system components.

R4: Security and Privacy of User Data

The security and privacy of user data must be guaranteed. Since there is likely to be an increase in the number of systems that continuously monitor context from various sensors or repositories, controlled access to data pertaining to the whereabouts of a user and their activities will be required for users to feel comfortable about making this personal information available [Brown,00b], [Schmidt,00]. For example, during an extension to the GUIDE prototype, tourists were able to share their location context with other users of the system to facilitate collaboration [Cheverst,00d]. This information can then be used, for example, by a tourist when trying to make a decision as to where to have lunch. Tourists are able to communicate with other tourists shown to be in (or who have previously been in) restaurants or cafés that day. This scenario highlights the requirement for suitable access control mechanisms or policy adaptation techniques [Efstratiou,00] in order to enable context use without compromising user privacy. Moreover, as Brown *et al* [Brown,00b] point out, each user must have control over information relating to themselves, and must be free at any time to change the accessibility of the information or the information itself. Furthermore, the ways in which an application uses information must be clear to users [Cheverst,00d].

R5: Extensibility

Applications aimed at operating in mobile distributed environments must be extensible in order to accommodate change and enable applications to adapt to new environments. Within the context of the GUIDE prototype, the introduction of new hardware, such as a new location mechanism, cannot be easily achieved. In more detail, the initial GUIDE prototype makes use the ORiNOCO wireless network for providing users with location information. At best, this provides relatively coarse

grain accuracy in terms of cells, that is, users are only able to determine which geographic region of the city they are located in. During a field trial evaluation of the GUIDE system [Cheverst,00a] it was discovered that at certain points of a city tour additional finer grained location information, such as data gathered from a GPS device, could help tourists with navigation. Furthermore, supplementing the infrastructure with indoor location technologies such as Active Badges or Bluetooth devices could enable tourists to continue using the system and follow internal tours of museums and other tourist landmarks [Davies,01].

These usage scenarios highlight a similar limitation to that found with the CyberGuide project [Long,96]: More specifically, a tight coupling exists between the GUIDE application and the hardware infrastructure on which it is based. Consequently, should an alternative location technology become available during an application session, such as a GPS or Bluetooth based service, the system is unable to re-bind or make use of this service.

To overcome this problem and allow the system to be extended more readily, a requirement exists to support a more general mechanism which removes the tight coupling that currently exists between applications and infrastructure.

R6: Modelling the Environment

As the move toward the ubiquitous computing vision first described by Weiser [Weiser,93] continues, the ability to facilitate complex interactions between users and device's will necessitate more than the devices capability to sense and adapt to contextual information. Improved user models (e.g. user profiles, social protocols) and environment models (e.g. location or geometric world models) will be required to supplement next generation *intelligent* applications [Cheverst,01c]. Work on geometric world modelling as part of the Microsoft EasyLiving project [Brumitt,00a], location models proposed by Leonhardt [Leonhardt,98] and Jose [Jose,01b], and the work on Active Spaces [Román,00] provide evidence of a successful shift toward ubiquitous computing. However, currently there exists no unified approach to world modelling and existing approaches suffer from similar drawbacks to those mentioned in requirement R5.

R7: Management of Shared and Distributed Data

The issues which pertain to information residency, retrieval and access to distributed data pose interesting challenges. First, in mobile environments which consist of both static and mobile elements there is a need to locate data (e.g. context, user data) and computational resources (e.g. services, functionality). For example, based on resource availability, a distributed system may choose to migrate some computation or data to a different processor [Gray,96]. This poses the question of where data should reside. In most cases this may depend upon the type of data in question, for example, sensed context is unlikely to reside on the device itself due to the physical constraints placed upon the device. In this case, proxies or services may be responsible for maintaining and controlling access to this type of data. A second question which arises in relation to user data is whether this information should be stored locally [Cheverst,99a] or stored in a central (trusted) repository and be accessed remotely by mobile clients using a secure channel [Schmid,01]. This may be particularly pertinent when considering user data, such as a working set of files, where the data is perhaps more sensitive to its user and a sense of ownership exists [Brown,00b].

Finally, when considering interactive or collaborative applications [Dix,99a] more dynamic distribution of data may be more appropriate and may require adaptive interaction models (requirement R3). The issue of data management is difficult to address since, in most cases, the method of data residency is determined by the specific application domain [Brown,00b]. For example, in the PARCTAB experiments, the tabs essentially act as user interfaces to remote data services since the indoor communications technology adopted provides good coverage. However in an environment with poor or variable communications coverage such an application may require locally cached data in order to offer adequate levels of service to its user [Davies,99].

R8: Configuration and Interoperability

Several factors influence the need to support resource discovery within mobile environments. First, as a user roams, mobile applications often have functionality distributed across an environment and frequent changes to the availability of these distributed entities is highly likely. Second, applications often exploit resources

within the immediate vicinity (local domain), for example, printers and network file spaces [Nelson,98]. Access to these resources are likely to be dynamic due to changes in the device that runs the application, particularly its physical location. It is therefore necessary for applications to automatically update service information when their context changes. Finally, from a user perspective, the ability for ad hoc discovery of other users, based upon context, increases the likelihood of *chance encounters* [Izadi,00a]. It is known that chance encounters, that is, the unscheduled meetings between people that occur in such places as corridors, are crucial to the functioning of organisations [Backhouse,92], [Eldridge,00].

A dynamic discovery mechanism should include support for all aspects of the discovery life cycle presented in chapter two, namely, a*nnouncement, discovery, description*, *configuration* and *interoperability*. Furthermore, discovery based solely upon physical location (used in current discovery technologies) may be a constraint in context-aware environments. Therefore, it is essential to support discovery based upon other forms of contextual information such as a user's interests or their current activity. A dynamic discovery mechanism with these capabilities is more general than current solutions (e.g. discovering a nearby printer) since it facilitates the discovery of contextual entities (i.e. users or services) based upon more than just physical location.

4.2.2.3 Requirements for Supporting Context-Awareness

R9: Context Capture

Chapter two described how location, orientation and identity have traditionally acted as the basis for a large number of context-aware applications [Abowd,99]. This informs a requirement to support a wider variety of contexts sensed from both the physical and virtual worlds. Physical context includes anything that can be sensed by hardware devices such as GPS devices [Garmin,01] or weather sensors [iButton,00b]. Virtual context refers to context obtained through the use of software components, for example, monitoring keyboard activity, processor load or documents currently open [Izadi,00b].

R10: Context Interpretation

For applications to successfully utilise context in a meaningful way, interpretation of context sensed from the environment may first be required. For example, consider an application that makes a request for a notification when a meeting is taking place [Dey,99b]. Location information could be sensed from the environment and interpreted to determine user identity or location and to check co-location. In addition, information such as user orientation and sound levels could be combined in order to determine that a group of people are in face-to-face contact and that a meeting is taking place. From an application developer's point of view, this interpretation should be transparent since a notification of when a meeting is taking place is all that is required. If an application developer is forced to handle the interpretation at the application layer the ease in which the interpretation is reusable by other applications is reduced. For example, a change in the underlying hardware used to gather context, such as the use of video for user recognition instead of orientation and sound, would require that each application would have to re-implement the context interpretation component.

R11: Infrastructure Transparency - Separation of Concerns

Mobile context-aware applications must be able to utilise a wide range of computing devices, heterogeneous communications and sensing technologies and offer services irrespective of the underlying infrastructure. The author believes that to a large extent the problems of tight coupling that currently exists between applications and the enabling technologies [Long,96], [Davies,98b] have had a limiting affect on the range contexts used within applications. Since software is written for sensors on an individual basis [Dey,99b] and often with no common structure between them, application designers wishing to make use of particular sensors employ *ad hoc* and application specific design techniques to integrate them into their applications. As a consequence this makes them both difficult to maintain and significantly reduces the ease to which new sensors (and therefore context) can be added. One possible outcome of this approach is that applications tend to use context in a limited way (e.g. just location and identity) at any one time. To facilitate a wider use of context (requirement R6) there needs to be a layer of abstraction between applications and underlying infrastructure [Izadi,00b]. The provision of a layer of abstraction enables

applications to be device, operating system and communications medium agnostic. Furthermore, the ease of which applications can be evolved or extended will be increased since changes to the underlying technologies will not adversely affect the operation of higher level applications or system components.

R12: Presentation, Adaptation and Persistence of Context

There are a number of ways in which contextual information can be exploited by an application. These include:

- **Presentation**: A common way to exploit contextual information is to process it in to a suitable form, perhaps by summarising the data, and displaying it to the end user. The reason for this approach is twofold. First, the context may be of interest to the end user, for example, the location of a friend that is currently out of the office. Second, the user may require the context in order to operate the system correctly. As [Dix,95] states *"people are very adaptable and, given suitable information about what is going on, they are often able to solve problems themselves*". To highlight this, consider a person carrying a mobile telephone; by providing feedback (i.e. awareness) relating to the current network status, users are able to appreciate that they may or may not be able to make/receive phone calls given their current context.
- **Adaptation**: Adaptation to context refers to situations where an application itself exploits the contextual information to determine the data services to provide to the end user. The adaptation approach is particularly beneficial to mobile applications, where applications must base their current semantics upon the user's context (often defined by the user's location and profile). The GUIDE system presented in chapter three demonstrates an instance of this. This system is of minimal use when it displays a global set of information since one could argue that a written document containing the same information is equally useful. However, the context sensitive nature of the systems offers considerable benefits since more precise tour content is possible based on end user location [Davies,98b].
- **Persistence**: Exploiting contextual information through persistence involves awareness information being gathered from the environment and stored for

later retrieval. Persistence can be exploited by applications themselves to *infer* new knowledge or establish trends. Returning to the tour guide application again, here, information regarding locations of users can be persistently stored and later utilised to determine the popular sites within the city and information that is often requested at those sites.

Table 4.1 categorises the systems and applications described in the previous chapter in terms of the context types supported and how that context is utilised. The context types presented include Activity, Identity, Location, and Time. For the use of context the three main features are described Presentation, automatic Execution, and storage for later Retrieval.

System	Context Type				Context Use		
	A	I	L	T	P	E	R
Active Badge Call Forwarding		X	X		X	X	
FLUMP			X		X		
CyberGuide		X	X		X		
Teleporting	X	X	X			X	
CyberDesk	X				X	X	
Audio Aura		X	X	X	X		
AROMA		X	X		X		
Stick-E Notes Guide		X	X	X	X		X
Stick-E Notes Reminders	X	X			X		X
Forget-Me-Not	X	X	X	X			X
Satchel		X	X		X		

Table 4.1 - Summary of context types and context use

This table illustrates that there is a requirement to support various types of context and that context use is necessary at many different levels.

R13: Ability To Support Awareness

The supply of context information (awareness) to users, applications and other system components is intrinsic to mobile and context-aware computing. Awareness can be described as the antithesis of transparency in that it is concerned with the supply of information to system components as opposed to the masking of it [Dey,99b]. For applications to adapt successfully *feedback* pertaining to the state of the operating environment is essential. An example of this is presented in MOST [Friday,99], where it is shown that by making users aware of the identity of group members experiencing poor communications QoS prevents group members from being forced

to make potentially false assumptions regarding the current state of connectivity with a group of collaborative users [Dourish,92].

Within the context of the GUIDE system, user preferences, location information and environmental factors such as weather predications and attraction availability act as forms of context. Sensing of these contextual attributes is distributed throughout the environment, for example, location updates are received locally through the `GuidePositionSensor` interface and context such as weather is gathered remotely and dispatched as events. The distributed nature of the system could be exploited further to provide users with awareness of other users of the system and perhaps their device capabilities. Consider the scenario in which a family group are visiting the city. Here, the parents of the family may be equally (if not more so) interested in knowing the whereabouts of their children in addition to their own location. A further requirement for supporting device awareness was derived when the GUIDE system was extended to allow collaboration between users using different mobile devices [Cheverst,01d]. Here, a user of the system who gains access through a WAP interface (see section 6.2.5.3) has access to similar functionality despite their device constraints (i.e. paying for communications using a mobile phone) does not originally have these constraints reflected at the user interface level. That is, two communicating users within this system would not be aware of the other users constraint's.

R14: Ability To Support Context Sharing Across Applications

The GUIDE prototype was originally designed as a stand alone mobile application, although during the user evaluation it became apparent that a user's context (user model) could be re-used or shared across a number of applications. Consider a visitor to the city who is hoping to visit a number of the city's attractions in addition to attending a business meeting in the city. There would be a clear benefit to the user if a range of applications (such as a calendar application) had access to the same environmental context. Thus, for example, events received through the GUIDE infrastructure relating to attraction availability or traffic announcements could be accessed by a calendar application and used to automatically re-schedule appointments or notify clients of any perceived delays.

Moreover, there exists a requirement for supporting shared access to application functionality in addition to sharing state, for example, the functionality afforded by the location sensing and the tour guide components found as part of the GUIDE prototype could easily be incorporated into other applications that may require location awareness and navigation services.

R15: Specification and Representation of Context

Future context-aware applications will make use of a wide array of computational, communication and sensing technologies. To enable applications to successfully evolve and make use of new technologies, applications must be easily maintainable and extensible (requirement R5). To achieve this, work on standardising common representations for context is required. This could be considered a natural extension to current efforts within the UPnP forum whose members are aiming for a similar goal and trying to establish standard device descriptions for a wide range of devices, particularly audio and video devices [Microsoft,99]. Within this domain, the agreement of common device descriptions and protocols between hardware manufacturers will enable applications and services to operate seamlessly across a number of environments (e.g. home audio/video networks).

The integration of different hardware devices is the first phase towards establishing ubiquitous or smart environments. To facilitate the integration between different context-aware systems, which is currently difficult or impossible, a similar approach is required. Context is currently represented and handled in a bespoke manner [Dey,99a]; however, Ryan [Ryan,99] has proposed ConteXtML, a simple XML based protocol for exchanging contextual information, field notes and map data between a mobile client and a server. Similarly, Byun *et al* [Byun,01], in an extension to the Lancaster GUIDE system, have recently proposed a two-layered XML DTD based approach for representing context, in order to support the interoperability of context between independently developed context-aware applications. This approach allows the flexible adoption of a separate DTD for each application domain as well as common definitions of context types that may be shared.

4.2.2.4 Overall Analysis

This section has described the key requirements for supporting context-awareness within mobile or distributed environments (summarised in table 4.2 below). The requirements have been established through an analysis of the related work detailed in chapter two and a critique of the GUIDE system presented in chapter three.

Requirement	Mobile Context Awareness
R1: Supporting User and Device Mobility	☺
R2: Support Persistence of Application and User State	☹
R3: Support Flexible Interaction Models	☹
R4: Security and Privacy of User Data	☹
R5: Extensibility	😐
R6: Modelling the Environment	☹
R7: Management of Shared and Distributed Data	☹
R8: Configuration and Interoperability	☹
R9: Context Capture	☺
R10: Context Interpretation	☺
R11: Infrastructure Transparency	☹
R12: Context Presentation, Adaptation and Persistence	☹
R13: Ability To Support Awareness	😐
R14: Ability To Support Context Sharing	☹
R15: Specification and Representation of Context	😐

☺ = full support. 😐 = some support. ☹ = no support.

Table 4.2 - Analysis of state of the art

The analysis has highlighted a number of limitations, which largely focus on the lack of application flexibility for supporting operation within heterogeneous environments and the inability to share context between applications. The requirements point towards the need for a more general approach to supporting context-aware application design better suited to mobile environments and one which reduces the coupling between applications and infrastructure. The primary goal of such an approach is application flexibility and, by employing a separation of concerns between applications and infrastructure, better support for operation in a rapidly changing execution environment can be achieved.

Furthermore, the related work presented in chapter two and table 4.1 can be used to show how, in general, there exists an overlap in the types of context utilised by applications, that is, traditionally, location, identity and activity form the basis for context-aware behaviour. However, there is currently little or no potential for

exploiting these commonalities to enable applications to share context or successfully operate in alternative environments or configurations, other than their initial target domain.

This observation shows that many context-aware applications *share common*, generic-application requirements, such as access to specific *types* of context or functionality. Therefore, an approach that provides shared access to low level services may benefit context-aware applications. Furthermore, in addition to generic requirements, applications are likely to have a set of application-specific requirements. As an example, consider a system that makes use of a location service for providing positional information to an application. A service based on GPS could clearly be easily re-used by several different applications, however, one instance of an application may require that longitude and latitude data be converted into an alternative format before use [Izadi,00].

The author believes that in order to withstand a fluctuating operating environment, mobile context-aware applications must be designed with mobility in mind [Demers,94]. To support this, a suitable access mechanism to the underlying infrastructure can be generalised in order for application developers to avoid the burden of designing applications with specific hardware or changing target domains in mind. By providing a suitable mechanism for enabling applications to explicitly state their contextual constraints (i.e. interests and attributes), more dynamic applications can be created that better suit the frequently changing environment in which they reside. Furthermore, by exploiting user and device awareness and allowing controlled access to context information, the sharing of context can be achieved which, in turn, facilitates the development of collaborative mobile applications.

Therefore, to summarise, by providing an appropriate layer of abstraction between applications and infrastructure, context-aware applications will have the benefit of increased support for user mobility, mobile awareness, context sharing and flexibility with respect to a heterogeneous processing environment. Furthermore, providing a more flexible approach and removing the burden of having to deal with a wide range of issues incidental to the development of applications, more applications can be realistically deployed and evaluated in settings outside of the research laboratory. Only through the rapid development and experimentation with context-aware systems

can exploration into issues relating to the use of these systems in *real* settings be established in order to further advance the research field in general.

4.3 A Context-Service Based Architecture

4.3.1 The Need For Context Services

Based on the requirements specified in the previous section, the remainder of this chapter argues the need for more appropriate, that is dynamic, support for context-aware applications within mobile environments, before detailing the design of a prototype implementation context-service based architecture. The analysis of the related work and critique of the GUIDE prototype implementation act as useful pointers towards identifying the support required by a context-aware architecture for use in mobile environments. More specifically, it is clear that any context-aware architecture must be capable of supporting the complex interactions that exist between users and the context sensing infrastructure affording 'intelligent' services. As a result of the complexity relating to the use of context, so increases the difficulty when maintaining and extending system functionality to include support for new forms of context.

Therefore, the primary role of any context-aware architecture must be that of simplification, i.e. simplification in terms of creating, maintaining and using context at the various levels (especially the application level and infrastructure level). By applying well understood object oriented principles to the development of context-aware applications and creating self-contained contextual entities, or services, with well defined interfaces, there exists an immediate increase in the ease in which components can be maintained or constructed [Franz,98].

Facilitating access to functionality through service interfaces and *pushing* these services into the infrastructure shifts much of the weight of context-aware computing into the network-accessible distributed environment [Hong,01]. Moreover, by creating uniform abstractions and reliable services for commonly used operations, service infrastructures make it easier to develop robust applications targeting a diverse and constantly changing set of end-systems.

In addition, an architecture based on context services simplifies the process of incrementally deploying new sensors, devices, or services. Furthermore, in terms of scalability, services should be able to scale up to support a potentially large user community [Davies,98a]. In addition, the use of services as an abstraction mechanism over the underlying technologies removes the burden of context acquisition from individual applications. Finally, this abstraction enables application designers to develop applications and supporting services independently since no direct relationship exists between application design and the infrastructure on which they reside.

To summarise, the notion of services is a well understood principle in distributed computing [Microsoft,01c] and one which suits the dynamic nature of mobile context-aware computing. Through the use of context services, a level of abstraction and flexibility is provided to application developers capable of supporting the following features: to be flexible enough to operate within a range of application domains, to support multiple applications accessing the infrastructure simultaneously, to reduce the burden of designers of context-aware applications, and to facilitate the use of arbitrary context types.

4.3.2 Introduction and Motivation

The approach detailed throughout the remainder of this thesis aims to achieve the goal of application flexibility for context-aware applications within mobile environments and support incremental development using a separation of concerns between context-aware applications and supporting infrastructure, as shown in figure 4.1. In essence, this separation of concerns can be used to provide a standard interface to the underlying infrastructure and to facilitate the sharing of context services between applications. As a result, application developers are able to more readily build, deploy and evolve complex dynamic applications with the following properties: to allow a user to work and move between devices during the course of a day, to maintain a consistent view of the state of their environment irrespective of device or application being used, and to include support for application adaptation and state management in relation to a changing execution environment involving the discovery and utilisation of new services.

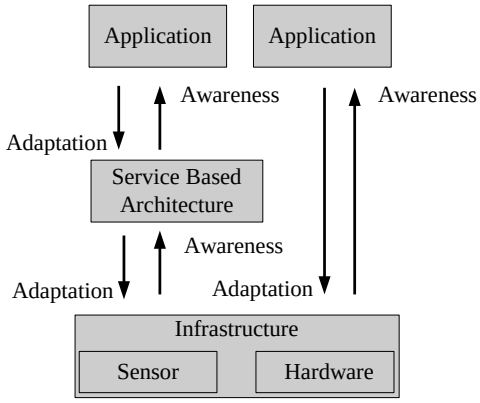

Figure 4.1 - Layer of abstraction

The approach shows how through the use of *context constraints*, *typing* and *discovery* the coupling between the application and infrastructure is reduced resulting in applications with greater flexibility. Furthermore, users are provided with greater levels of control. For example, consider a scenario in which several similar services (e.g. location) each with different properties (e.g. accuracies, power constraints) are available. In this case users may decide which service to use based on their current context.

The remainder of this chapter defines the properties of the service based architecture to support mobile context-aware applications. In particular, the major concerns for the research presented in this thesis are:

- **Flexibility**: To enable applications to adapt to use across a wide range of heterogeneous devices and execution environments, that is, independent of the underlying infrastructure or physical environment.
- **Evolution**: To aid application evolution and extensibility by using a separation of concerns between using context and the underlying infrastructure.
- **Partial Connectivity**: To enable context-aware applications to operate successfully in mobile environments even during periods of disconnected operation and to make use of techniques such as dynamic discovery to utilise new context services.
- **Context Awareness**: To build upon the notion of mobile awareness [Cheverst,99b] and facilitate the awareness of users, devices and services across a range of heterogeneous devices and mobile environments and to

support the sharing of contextual sources between application processes and instances.

4.3.3 A Context-Service Based Architecture: Overall Approach

The overall approach is based on the system architecture shown below (see figure 4.2). It is assumed that a fixed backbone network model is present that connects a number of interconnected cells or sub-networks, that is, a trusted network infrastructure consisting of fixed networks and hosts (fixed or mobile) with reliable communications via the backbone network. For the purposes of this thesis, self organising or ad hoc networks are not considered [Hodes,97]. These assumptions enable the communications between system entities to be simplified. In essence, a number of mobile nodes may exist within a wired or wireless sub-network. Access to the context service architecture gained through the network involves a context server. Furthermore, the central context repository acts as a trusted and reliable repository for all information (state) relating to applications and their sessions (see section 5.2.1). More specifically, the central context repository acts as remote storage where data may reside after an application has terminated.

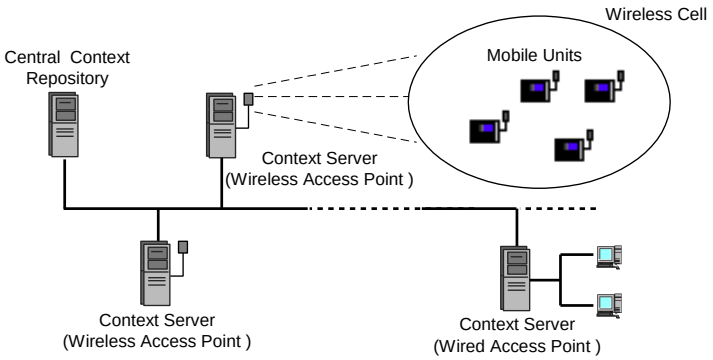

Figure 4.2 - Overall system architecture

In terms of the software components, the system consists of two main entities, shown in figure 4.3. First, there is the context service provider (CSP) which in turn makes use of any number of lower level context services (abstract, translation or bespoke). The CSP acts as an application agent and is responsible for managing context pertaining to executing applications on that device, the discovery of new context services and the management of state to support operation during periods of disconnected operation.

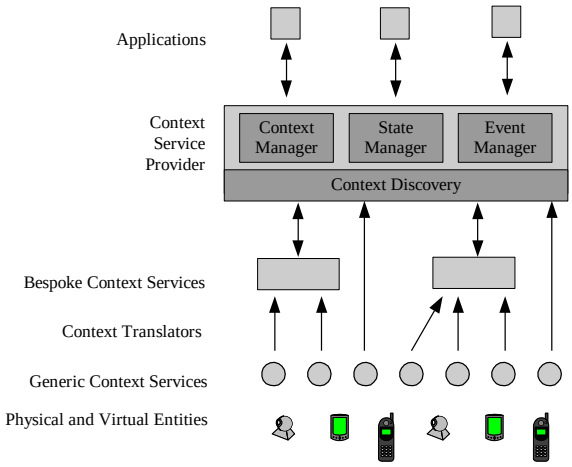

Figure 4.3 - The components of a service based architecture

The following section formally introduces the components shown in figure 4.3 beginning with the lower level generic context services. The chapter concludes by describing in detail the role of the higher level context service provider (CSP) and its constituent entities the context discoverer and the context, state and event managers.

4.3.4 Context Services

4.3.4.1 Overview

A context service represents any entity that is capable of providing contextual information to the rest of the system and represents an abstraction over the underlying infrastructure, which may consist of both physical and virtual entities. Physical entities represent any hardware device or sensor that provides context to a system component, for example, a temperature sensor. Virtual entities represent software components such as keyboard or power consumption monitors [Efstratiou,00]. In essence, these are bespoke entities with ill-defined interfaces and no standard communications protocol. The requirement for supporting the re-use of context services led to the notion of supporting both generic and application specific services (see section 4.2.2.4).

4.3.4.2 Abstract Context Services

Abstract context services provide system components with well defined interfaces and standard communications mechanisms to the underlying physical and virtual entities. In essence, abstract services act as wrappers to the underlying infrastructure and provide a way of hiding the specific implementation details from application developers. This allows developers to use a service, such as retrieving information from a sensor, without being overly concerned with how the data is collected from its source. Furthermore, abstract services include a number of context management features including storing the services' current and previous context (state), mechanisms for receiving subscription requests for contextual events and being able to respond to queries for context state.

To highlight the usefulness of abstract services, consider a scenario in which an application developer wishes to use a location device, for example the Garmin E-Trax Venture [Garmin,01], to provide location awareness to an application. Through the provision of a well defined interface allowing access to the state of the device, for instance its current latitude and longitude co-ordinates, an application developer is not burdened with having to know or realise a specific implementation in order to communicate with the device (i.e. reading streams of data from a serial port and interpreting them). By creating an abstract service and hiding the specific implementation details an application developer can utilise the service more readily. Furthermore, should the specific hardware device or method of context capture be updated, no further effort is required by the developer since the service interface will remain the same.

Abstract services offer two additional features. First, a service is able to store (make persistent) context pertaining to that particular entity's history, perhaps for a later retrieval [Spiteri,98]. For example, an abstract service representing a thermometer could be used to determine the current temperature reading but may also be able to offer a context-aware system the average temperature reading given a specific time interval. Second, an application may subscribe to an abstract service for context updates and avoid the need for periodic querying of a service's state. By subscribing to a service for events, context updates are pushed out to all interested parties. Abstract services therefore represent components that can be re-used and shared

between applications. Figure 4.4 shows a number of example abstract services used within the GUIDE II prototype detailed in chapter six.

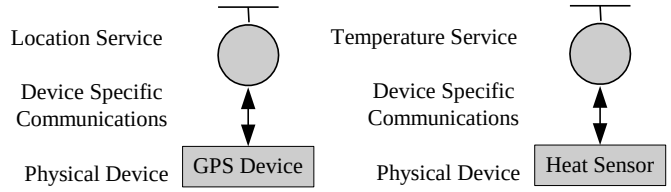

Figure 4.4 - Abstract context services

4.3.4.3 Context Translation Services

Context translation services enable context to be transformed from, generally, lower level (generic) context to higher level or more specific context. Consider the GUIDE system detailed in chapter three as an example. This prototype makes use of beacons to notify users of their location. However, the accuracy achieved is largely dependent upon the city's geography and more importantly on the location of the cell servers. This means that users are only able to determine their location in terms of their current region of the city and not in terms of specific locations within that region. During the development of the GUIDE II prototype [Cheverst,01d] (detailed in chapter six) the approach based on location beacons was supplemented with a GPS approach in order to provide more accurate positional information to users. This inclusion of this functionality would normally require large modifications to be made to the original GUIDE implementation to convert longitude and latitude co-ordinates into city landmarks. However, by using a translation service for this task the prototype was able to continue operating in its original manner while making use of the finer grained location information and without a large re-write of the client application. Furthermore, since the translation service is not actually part of the GUIDE application its functionality may be re-used by any number of additional components. Figure 4.5 illustrates the GUIDE GPS translation service based on the city centre of Lancaster.

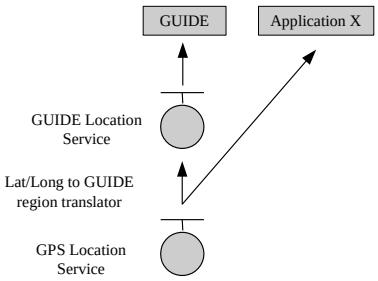

Figure 4.5 - Context translation services

4.3.4.4 Bespoke Context Services

The notion of bespoke context services enables application developers to make use of a variety of abstract and translation services to create specific functionality based on their target application domain. For example, Cheverst *et al* [Cheverst,01a] describe a service used to predict the likelihood of there being an impressive sunset over Morecambe Bay on a particular day. Since Lancaster is located only four miles from the west coast there are often spectacular sunsets in the bay area visible from a variety of vantage points in Lancaster, namely Castle Hill and Williamson Park. The virtual service was developed to predict the likelihood of there being a sunset based on a variety of factors. The factors considered to calculate the probability of there being a sunset included a user's location, their preferences, the time of day, current temperature and weather information. If a high probability was ascertained then users were notified and given the chance to visit the suggested vantage point. Figure 4.6 illustrates how a bespoke service may comprise of a number of abstract or translation services.

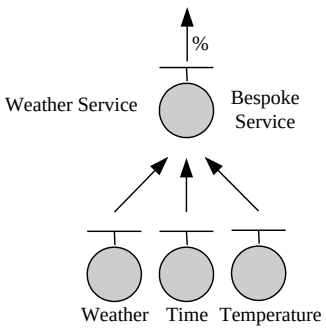

Figure 4.6 - A prototype context service

4.3.5 Context Service Provider

The Context Service Provider or CSP can be regarded as an application's agent or session manager and is responsible for providing a number of transparent context management features to higher level applications. The main components of the CSP are the context manager (CM), event manager (EM), session or state manager (SM) and context discovery component, which are introduced in the following section.

4.3.5.1 Context Manager

The context manager is responsible for managing an application's contextual interest and associated constraints. Applications register with the context manager specifying their context requirements, that is, their user stipulated context constraints (see appendix A) including required context types, security and privacy requirements. For example, an application may require the use of a location context with an accuracy of 10m. After registration the context manager locates all the services that offer similar features and ensures a match is found based on the constraints supplied. The associated constraints include privacy and security primitives that can be used to control how context is shared amongst applications. For example, within the context of the GUIDE prototype there was a requirement to enable users of the system to obtain location information relating to other GUIDE users, such as a teacher being able to locate their pupils within the city. In addition, privacy constraints were included so that users could be visible whilst remaining anonymous or be made invisible by the system if they did not wish to be located at all. These mechanisms provide a sound basis for exploring social awareness within mobile environments and more importantly allow a user to maintain control over information relating to themselves [Brown,00b].

4.3.5.2 Event Manager

The event manager is responsible for managing client subscriptions for contextual events or updates. Applications may use either a query based pull or subscriber based push mechanism for retrieving context from the environment. For example, a tourist with an interest in a specific landmark, such as Lancaster Castle, could register for updates ensuring that changes in context relating to that attraction are received by the client application. The event manager maintains a handle to all the user's interested

services and is able to discover and utilise new services based on the user's context constraints (i.e. the types and attributes). The communications protocol used for this component is based upon a modified and more general version of the broadcast protocol described in section 3.3.3.

4.3.5.3 State Manager

The state manager is responsible for maintaining a consistent view of the operating environment on behalf of client applications and is designed to overcome some of the limitations presented in section 4.2. In more detail, the two important functions supported by the state manager are session tracking and cache management.

Resources such as files or objects are stored locally (i.e. cached) during an application's lifetime and made persistent after an application session has ended by storing all data items in the context repository. This enables applications to successfully operate during periods of disconnection using the local cache and allows applications to detect the state of the environment once initialised from the persistent store, that is, the central context repository. Furthermore, a particular resource and its associated context can be used to aid cache management, that is, context-based caching. In more detail, resources include context such as timestamps and expiry information in order to define resource mutability and aid cache management. For example, within the GUIDE application the time sensitivity of a resource varies widely. A hypertext page relating to the history of a particular landmark may include a timestamp indicating that this data item is unlikely to change frequently and can be cached locally. Similarly, a hypertext page relating to a café's menu may change on a regular basis (providing different breakfast, lunch, evening meal menus) and may therefore be less useful to cache given that the resource is likely to be updated on a regular basis.

4.3.6 Summary

This chapter has provided an analysis of the related work detailed in chapter two and a critique of the GUIDE prototype implementation detailed in chapter three in order to establish the architectural requirements for supporting mobile context-aware applications. Following on from these requirements, the chapter then described the

need for a context service based architecture and detailed how this mechanism provides a solution capable of satisfying the identified requirements. Furthermore, the architectural components of a prototype context service-based architecture were introduced. This architecture has been designed to overcome the limitations derived through an analysis of the GUIDE system whilst, simultaneously, building upon the positive features of this complex application as described in chapter three. In essence, the architecture focuses on providing a level of abstraction between a context-aware application and the context infrastructure. As a result, this mechanism provides enhanced support for controlling and managing access to lower level context services, in addition to implicit support for user mobility, by way of context-based caching. The design included a description of the constituent components which can be used together to support context-aware applications designed for dynamic execution environments. The following chapter details the implementation of the context service architecture and, in addition, describes several prototype context-aware applications developed using the architecture.

Chapter 5

Architectural Design and Implementation of a Context Service Based Architecture

5.1 Introduction

The previous chapter demonstrated the need for a context service based architecture and introduced the key architectural components of a prototype context service designed to provide support for mobile context-aware applications. The prototype context service introduced demonstrates a more flexible and dynamic approach to context-aware application design and provides implicit support for application operation within dynamic environments. Moreover, the context service is designed to provide the application programmer with a convenient mechanism for supporting mobility and application extensibility through an appropriate layer of abstraction between applications and infrastructure.

This chapter considers the context service from an engineering perspective and describes the prototype implementation in some detail including the associated protocols used in its construction. The chapter also details a number of context-aware

applications designed using the architecture as an initial illustration of the service in use. Further studies detailing the infrastructure in use by the GUIDE system are carried out and detailed in the evaluation chapter.

5.2 Design and Implementation of a Context Service

5.2.1 Context Service Provider

5.2.1.1 Introduction

To support the dynamic nature of mobile environments, the context service architecture is designed to aid applications utilise context efficiently whilst simultaneously offering support for mobility. In more detail, applications may register their contextual interests and constraints with the context service provider and allow the various components to control an application session on their behalf. Furthermore, applications may utilise the session management features to support operation during periods of disconnection. To allow for flexibility and to support operation in dynamic environments, application developers may decide at design time which services to utilise without explicitly binding an application to a particular technology. For example, an application which makes use of location as a form of context could be designed to register with the supporting architecture and specify location and any associated constraints, such as timeliness and accuracy. This approach removes coupling between applications and infrastructure and allows the architecture to take responsibility for determining the location services to utilise at run time. Furthermore, in a rapidly changing mobile environment in which a number of location services may be present during an application's lifetime, the burden of dynamic discovery does not lie with the application developer but with the supporting infrastructure.

5.2.1.2 Accessing the Context Service Provider

Applications wishing to utilise the services offered by the context service provider must first go through a simple registration process, summarised in figure 5.1. Since the CSP may serve more than one application simultaneously, each individual application must supply its credentials in order to utilise the services offered by the

CSP. The registration process provides applications with an explicit mechanism to stipulate any application specific context that may be used in order to tailor the services offered. In more detail, applications must provide the following: application name, a session identifier, a username and password pair for the user, the host name and an available port number on which to receive contextual events (call-backs), context types of interest and any associated constraints. For example, context types such as camera, location or printer may have the associated constraints resolution, accuracy and pages per minute respectively.

Once this personal information is submitted to the CSP, the details are processed via the state manager which returns two data items to the registering application: a dynamically generated unique identifier and a port number. The port number must be used by the client for all further communication with the CSP and the unique identifier must also be supplied in further communications to allow the CSP to maintain state pertaining to each individual application session, i.e. a session log.

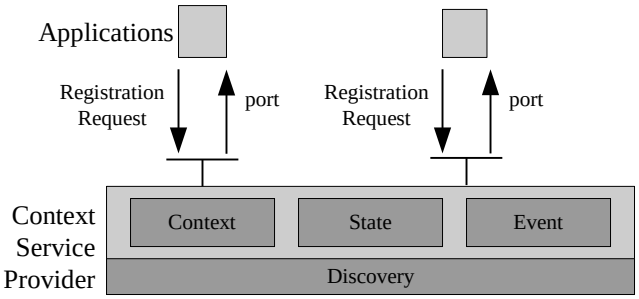

Figure 5.1 – The Context registration process

All future communications between an application and the CSP involve the use of sequence numbers (or event identifiers) which are stored locally by the session manager, a feature which appears transparent to the users. This facilitates support for disconnected operation since a session log of all interaction is created and can be used for synchronisation purposes following disconnection from the network, discussed in more detail in section 5.2.6.2.

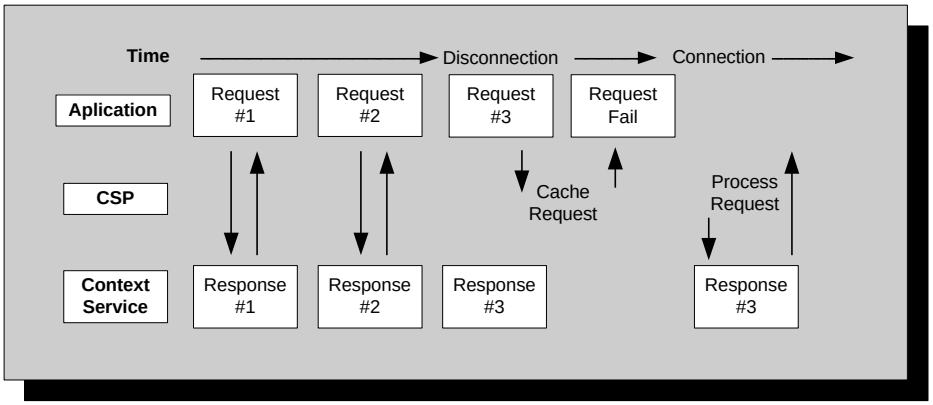

Figure 5.2 - Session tracking using sequence numbers

The session manager is able to use the unique session identifier supplied during registration along with the username and password pair to check the local cache for any persistently stored user state, such as files or objects from the last application session, that may need to be resumed. If no state is found locally, a request may be made of the context server (i.e. local cell server or access point) which may in turn query the central context repository to retrieve any previous application state.

The context types specified by the client application are passed to the discovery component (see section 5.2.3) which searches for contextual services and retrieves handles to the relevant services based on the specified constraints. Figure 5.3 shows an example relating to the GUIDE II prototype (detailed in chapter six) in which the system detects the availability of two alternative services of the same type. In this scenario, the constraints specified by the user are used to aid the system determine the most appropriate service based on the user's current context.

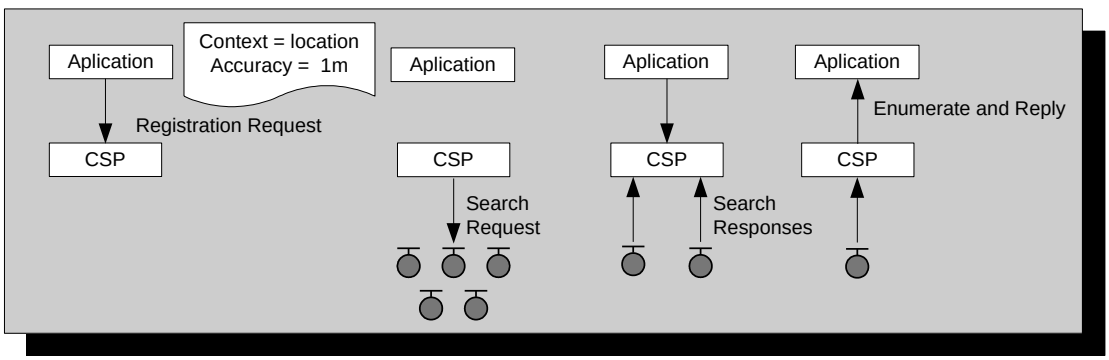

Figure 5.3 - Discovering multiple services

The session manager's primary responsibility is to maintain state on behalf of an application. Therefore, if during an application session additional services are discovered, new constraints are specified, or existing constraints are modified the session manager is responsible for ensuring that the factors are considered in order to guarantee the most appropriate service is available and in use. The session manager maintains a list of all application to service mappings and uses this data to ensure that a consistent view of the environment is maintained.

The `ContextServiceProvider` interface provides the operations shown below in figure 5.4.

```
public interface ContextServiceProvider
{

/* This method allows applications to login to the
 * platform and retrieve a unique session id and port number
 */
public DataPacket login (){ ... }

/* This method allows applications to register with the architecture
 */
public Boolean subscribe(String name) { ... }

/* This method allows applications to unsubscribe with the architecture
 */
public Boolean unSubscribe(String name) { ... }

/* This method causes the state associated with a
 * particular application session to be made persistent
 */
public Boolean save (Context c) { ... }

/* This method causes the state associated with a
 * particular application session to be restored
 */
public Context restore (String name, String password, int id) { ... }

/* This method is used by an application to indicate that it
 * wishes to terminate and thus store its state and unsubscribe
 * from the platform.
 */
public Boolean logout(String uName, String pWord) { ... }

/* This method allows an application to update any
 * contextual preference
 */
public Boolean update(Context C) { ... }

}
```

Figure 5.4 - The context service provider (CSP) API

5.2.2 Context Services

Both abstract and bespoke services act as wrappers to the underlying infrastructure and provide a way of hiding the specific implementation details from the application developer. Their functionality is provided by the `ContextService` interface which includes specific implementation details in addition to the generic properties inherited from the abstract `BaseContextService` interface. This is used as a common interface to allow standard methods to be defined. Furthermore, sub-classes are able to overwrite these standard methods should extra functionality be required. This approach offers the advantages such as flexibility, extensibility and reusability as described in section 3.4.2.2.

```
public abstract class BaseContextService
{
/* Returns a Vector which can be enumerated by higher levels to
 * determine the state variables represented. E.g. latitude, longitude
 */
public Vector getState() { ... }

/* Returns a Vector which can be enumerated by higher levels to
 * determine the services (operations) available.
 * E.g. Playing or stopping a device.
 */
public Vector getServices() { ... }

// Return the current state (context) for a particular service attribute.
public Context getContext() { ... }

/* A query for a particular piece of Context which returns an
 * object based on the result of the query.  E.g. get("id"); would cause
 * the service to return its handle.
 */
public String get(String name) { ... }

/* To enable an application to be brought up to state the replay method
 * is used to replay events between a specified time interval or
 * some specific conditional context attribute.
 */
public Vector replay(GregorianCalendar from, GregorianCalendar to) { ... }

public Vector replay(Context C) { ... }

/* A call to this method causes a service to return a XML File
 * description containing details of its service interface.
 * i.e. service attributes, operations and communications channel.
 */
public File getDescription() { ... }

}
```

Figure 5.5 - The interface specification for a BaseContextService

One example of a context service might be a service representing a weather sensor, that is, an entity collating and providing weather related context to applications. This

service may include state representing the current state of the environment, for example, current wind speed, humidity, temperature, etc. The `BaseContextService` interface includes access methods which allow the context associated with that service to be queried by applications within the system. The `BaseContextService` interface contains the methods shown in figure 5.5.

```
public Context getContext();
```

This method is used to retrieve a service's current context. The result of the `getContext` method is an object of type `Context` containing the information for the requested service. This interface also includes a generic get method to enable specific data items to be retrieved. For example, a call to `get("temperature");` would cause the service to return the value relating to the current state of that attribute.

```
public File getDescription();
```

This method is used to retrieve a service description relating to a context service. Using this description, the properties (i.e. state variables and access methods) of the context service can be determined by an application. Once the service interface is known, the individual data items can be queried using the aforementioned generic get method.

The `ContextService` interface enables a specific instance of a generic `BaseContextService` object to be created. Once instantiated a thread of execution begins and a running object (i.e. a context service) has two primary responsibilities.

- **Advertisements**: A context service begins announcing its presence by periodically multicasting service discovery packets. The discovery mechanism used is described in more detail in section 5.2.3. Clients may register with a service for call backs if the service is of interest, for example, a service that provides the location of people within an office building obtained from a variety of sensors may be subscribed to in order to be notified of a particular user's location.
- **Queries**: A service is also responsible for responding to explicit requests for data, context or subscriptions from applications, for example, an application

may a periodically request the current context for a specific service, such as the current temperature reading from a temperature sensor.

The interface to the context service component is shown in figure 5.6.

```
public class ContextService extends BaseContextService implements Runnable
{

/* This method causes a thread of execution for the service to
 * begin and by default starts periodic service advertisements
 * before listening for incoming requests from higher level applications
 * and lower level hardware (such as a sensor).
 */
public void run() { ... }

/* This method commences the service announcements based on the specified
 * delay (between announcements). A delay of 0 (zero) halts service
 * announcements.
*/
public void advertise(int delay) { ... }


/* Returns a Vector which can be enumerated by higher levels to determine
 * the services (operations) available. E.g. Playing or stopping a device.
 */
public Vector subscribe(host, port, appId, eventId) { ... }

/* A query for a particular piece of Context which returns an object
 * based on the result of the query.  E.g. get("id"); would cause the
 * service to return its handle.
 */
public boolean unsubscribe(host, port, appId, eventId) { ... }

}
```

Figure 5.6 - The context service API

The `run()` method is used to create a specific instance of a context service. Once instantiated, in addition to the service specific actions and communications for which the service is responsible (i.e. collecting data from a sensor), a service will periodically announce its presence according to the specified delay. A context service may be instantiated without advertising its presence to the rest of the operating environment by specifying zero as the delay parameter when created. A separate thread of execution is also started which listens for incoming requests, subscriptions and un-subscriptions from client applications.

5.2.3 Context Discovery

5.2.3.1 Overview

The context discovery mechanism is designed to provide a simple and fast mechanism for enabling applications to be made aware of new context services from within the environment. Since the development of a scalable discovery mechanism designed for mobile environments is beyond the scope of this thesis and not of direct concern to the author it is detailed as an area for future work in section 6.3. In addition, since the discovery mechanism is a self contained entity, it is generic enough to be extended in order to operate with any of the existing standards based resource discovery protocols detailed in chapter two. Moreover, the discovery mechanism presented below includes some features present within the Universal Plug and Play specification described in chapter two [Microsoft,99] and, in essence, provides a simple mechanism for context-aware applications to discover the range of context services on offer and to *automatically* select the most appropriate service for use. The approach described below, based on automatic selection of context services, is in contrast to the approach adopted by the UPnP forum which places a large burden on the *end user* to determine the semantics for any specific discovered service. In more detail, a large proportion of the communications that exists between a client application and a resource is related to the presentation of the service interface described in XML [Friday,01a]. Therefore, in terms of the work detailed in this thesis, the provision of a suitable discovery mechanism capable of determining the most suitable service based on the user stipulated context constraints is to be addressed and hence, the communications overhead associated with the presentation of service interfaces is irrelevant. The discovery mechanism detailed below can therefore be regarded as 'black box' providing the following two basic features:

- **Searching**: Requests for a service are made by the context service provider and services offering the required functionality must respond appropriately.
- **Announcements**: Services periodically multicast, to a well known channel their service announcements.

5.2.3.2 Context Discovery: Searching

Search Request

The context service provider is responsible for searching for context services of interest within the networked environment. To do this, a search message (a UDP datagram packet) containing the search criteria is multicast to a well known multicast address [Meyer,98]. Context services listening for search requests respond with a unicast message if the context types searched for are supported by the service.

A `DiscoveryPacket` multicast to the network must contain the following fields:

- **Message Type**: The message type must be set to SEARCH to identify the packet as a search packet.
- **Service Type**: The context service type of interest must also be specified, for example, a location service.
- **Delay**: Context services responding to search requests should delay their response by a random delay between 0 and the value specified (as an integer) by this field. This should aid the load balancing for the context service provider when processing incoming responses.
- **Client Identifier**: A unique identifier representing the client making the request.

Search Response

Any context service hearing a search request and whose service type matches with the specified in the search request must respond with a unicast `DiscoveryPacket` which must contain the following fields:

- **Message Type**: The message type must be set to RESPONSE to identify the packet as a response to a search request.
- **Service Identifier**: The service identifier contains a unique identifier relating to the service.
- **Service Type**: The context service type must also be specified, for example, a temperature service.

- **Timestamp**: Context services should insert a timestamp.
- **Expires**: The time in seconds from the time specified in the timestamp field for the validity of this service should be specified.
- **Service Channel**: A port number that can be used to query the state of the context service.
- **Event Channel**: A port number that can be used to subscribe to events.
- **HTTP Channel**: A URL for the XML description of the context service enables a client to access the context service interface via the HTTP protocol. See section 5.2.4. for more information on context service specification
- **Interface Description**: A description of the context service interface as a Java `ContextService` object. A handle to the specified service is now maintained by the context service provider.

5.2.3.3 Context Discovery: Announcements

The discovery protocol allows context services to advertise its services to the context service provider. It does this by multicasting discovery messages to a well known multicast group (address and port number). The format of the discovery message is similar to that of a response to a search request, with the difference that the message type is set to ANNOUNCE. The expires field notifies interested parties of the validity of the service and is normally set to expire at the same interval as the rate of sending discovery message, for example, a service that advertises its services every 60 seconds will include an expiration time of 60 seconds with all its discovery messages.

5.2.3.4 Determining an Appropriate Context Service

An application making a request for a specific context-service type may result in the CSP discovering several services of the same type (e.g. location, camera). In this situation, the services are enumerated and, according to the user stipulated context constraints, a service is selected which best meets the criteria specified by an application.

More specifically, the service selection process is determined by a combination of the logical operators detailed in Appendix B when applied to both the user stipulated

context constraints and the service description interface. For example, a user specifying that they require the use of a location context service may also specify constraints such as the use of a service capable of providing positional information accurate to less than 20 metres. To meet these requirements, when selecting a service the CSP will dynamically chose a service by interpreting the interface and searching for attributes matching the user requirements, as shown previously in figure 5.3.

5.2.4 Specifying and Using Context

The author believes there is scope for providing a general mechanism for specifying context built around the notion of context types. Context types are used to broadly define the particular category of context being represented. By creating types and templates for the specification of context, the sharing of context between applications can be achieved [Byun,01].

To highlight the potential usefulness of this approach consider the context-aware applications presented in chapter two which can be broadly classified into the broad domains shown in table 5.1. It is shown how within each domain the range of context types employed remains relatively static. This suggests that a simple mechanism to enable context representation within these domains could provide scope for context sharing between applications. For example, applications such as Forget-Me-Not and CybreMinder may offer benefits to their users by sharing contextual information.

Domain	Context-Aware Applications	Contexts
Guide Systems	GUIDE [Davies,98a], Cyberguide [Long,96], HIPPIE [Oppermann,98], [Oppermann,99b], Conference Assistant [Dey,99a] and C-MAP [Fels,98]	location, time, preferences, schedule, network connectivity, etc.
Office Assistants	CyberDesk [Dey,97], PARCTAB [Want,95], Stick-E-Note [Brown,96]	document context, physical parameters, proximity relationships, etc
Personal Assistants	Forget-me-not [Lamming,94], Remembrance Agent [Rhodes,96], ComMotion [Marmasse,00], Shopping Jacket [Randell,00] and CyberMinder [Dey,00a]	time, location, events, user tasks and goals, etc
Field Work	Adtrantz [Siewiorek,98] and FieldNote [Pascoe,98b]	sensor based context, e.g. temperature, etc
Intelligent Environments	EasyLiving [Brummitt,00a], Cooltown [Kindberg,00], TEA [Chen,99] and Adaptive PDA [Schmidt,98]	physical environment context relating to users and devices

Table 5.1 - A summary of context-aware applications

The approach described below was established during the implementation phase of a number of simple context aware applications, namely UbiChat and the digital In/Out board (see section 5.3.3) and during the re-engineering of the GUIDE system (see section 6.2). The design of these simple applications aimed to exploit and build upon the functionality provided by the GUIDE system (namely the use of location awareness) and provided the motivation for devising a more general approach to context specification (or representation) supported by the GUIDE tags introduced in chapter three. The GUIDE tags employ a simple mark-up language to identify context-sensitive aspects of hypertext pages. By extending the range and syntax of the tags employed a simple mark-up language specified in XML is capable of providing descriptions of contextual entities within a context-aware environment.

Figure 5.7 shows a sample context service template specified in XML used to represent the functionality of a GPS device.

```
<?xml version="1.0"?>
<context-service>
    <contextServiceType>Location</contetxSserviceType>
    <contextServiceId>GPS-Device-Identifier</contextServiceType>
    <contextServiceChannel>
       <host>name</host>
       <port>6000</port>
    </contextServiceChannel>
    <eventChannel>
       <host>name</host>
       <port>6100</port>
    </eventChannel>
    <serviceScope>
       <groups>
           <groupName>DMRG</groupName>
           <groupName>Ubicomp</groupName>
       </groups>
    </serviceScope>
       <contextServiceStateTable>
           <contexStateVariable>
              <name>positionX</name>
              <dataType>number</dataType>
              <allowedValueRange>
                 <minimum>0</minimum>
                 <maximum>1000</maximum>
                 <step>1</step>
              </allowedValueRange>
           </contextStateVariable>
           <contextStateVariable>
              <name>positionY</name>
              <dataType>number</dataType>
              <allowedValueRange>
                  <minimum>0</minimum>
                  <maximum>1000</maximum>
                  <step>1</step>
              </allowedValueRange>
           </contextStateVariable>
       </contextServiceStateTable>
```

```
       <actionList>
           <action>
              <name>setPositionX</name>
              <paramterList>
                     <paramter>pos</paramters>
                     <paramterType>long</paramterType>
              <returnType>
                     <Boolean>
              </returnType>
           </action>
       </actionList>
</context-service>
```

Figure 5.7 - A sample XML specification for a context service

The main components of the service template presented above are as follows:

- **Context Service type and identifier**: The context service type is used to identify the nature of the service offered, such as a printer service. The context service identifier is used to uniquely identify a specific service since more than one service of a particular type may exist.
- **Context Service state table and variables**: The context service state table is used to represent a structure of state variables in use by the context service. In essence, the state variable is used to represent any item of context (state) that can be queried by client applications. A state variable includes a name, the data type represented and the permitted range of values, if relevant.
- **Action Lists and Actions**: An action list is used to represent the possible actions or methods supported by a context service. An action includes a name, any associated parameters and types and the return type expected.
- **Service Scope**: Access to a context service may be controlled by specifying the service scope. More precisely, a service may be intended for a specific user and may therefore only be accessible by that user. For example, a service representing a user's personal diary may have restrictions placed upon the service, allowing only the creator of the service access, i.e. a private service. Furthermore, a service may include group access, for example, enabling all members of a department to access a personal diary. Finally, a context service may be visible to all system entities, that is, it be publicly available, such as a context service representing a weather sensor [Cheverst,00d].

A context service identifier distinguishes uniquely a particular service and is a common feature in resource discovery architectures [Microsoft,99], [Sun,99b]. The

identifier provides a simple way of distinguishing services uniquely despite the fact that several services of the same type may co-exist.

The notion of service type is used by the context-based architecture to support the automatic selection of context services based on specific functionality. Type definitions are used to inform applications of the semantics of the operational interface provided by a context service. For example, a context service of type Location will have different operational properties to a service representing a camera. The use of types in the context service discovery process provides simplicity and flexibility when performing type matching of search requests.

For evaluation purposes (see chapter six) the context service types supported by the current implementation include location, camera, network, weather, user, message, profile, tour and temperature. Furthermore, an XML template also exists for the user model (i.e. the GUIDE user profile object) detailed in chapter three. More specifically, the attributes relating to a GUIDE user profile object detailed in chapter three and appendix A are publicly accessible through an XML interface in order to be utilised from a range of context-aware applications.

5.2.5 Context Translation

The architecture provides a translation service that enables XML documents to be parsed and converted into Java objects and vice-versa. For example, the following sample XML document represents part of a user's profile. This structure can be interpreted by the system and converted into an instance of the `UserProfile` object in order for a GUIDE client application to read, manipulate and store the user's profile. Access to the user profile for example, through a WAP interface, is provided by accessing and modifying the XML document directly. In this instance the WAP interface gains access through a simple HTTP proxy server.

```
<application>
       <name>GUIDE Browser</name>
       <id>gb1</id>
       <host>mitchell.lancs.ac.uk</host>
       <port>3000</port>
       <active>true</active>
       <context>
              <type>location</type>
              <requirements>
                     <name>accuracy</name>
                     <value>200</value>
              </requirements>
              <privacy>
                     <access = all>
              </privacy>
       </context>
</application>
```

Figure 5.8 - A sample XML description for the GUIDE application

The advantage of using XML for context representation is that it does not force context-aware application designers to implement context services in any specific programming language. Instead, clients and services can be implemented in any programming language with the only requirement that the communications protocol and context specification be adhered to. However, all prototype implementations detailed in this thesis have been developed using the Java programming language.

5.2.6 Communications

5.2.6.1 Introduction

Chapter three (section 3.3.3) described an approach to providing support for a potentially large user community through data dissemination based on the broadcasting of information. Furthermore, it was also described how clients are able to issue specific requests for resources not contained within the broadcast schedule. This technique supplements the broadcast based approach with a pull-based approach. However, despite these two alternatives, it is possible for clients to operate using stale information or for information broadcast by servers to be missed by clients (for example if they leave a cell and only receive part of a broadcast transmission).

5.2.6.2 Engineering Issues

In order to provide a more reliable means of communications and allow applications to have where possible the most relevant data, the communications mechanism described in section 3.3.4 was extended to allow applications to subscribe to specific

context sources. More specifically, the broadcast approach supplemented with client pull was extended to enable applications to subscribe to particular context events. The communications protocol shown in figure 3.4 was extended to include the following additional types :

- **Subscribe**: The subscribe type is used when a client application wishes to subscribe to a specific item of context, such as the temperature of a particular sensor. By doing this, any changes to the state of the temperature will be propagated to all interested parties. A service that receives a subscription request is required to maintain a handle to the requesting client.
- **Unsubscribe**: Applications wishing to unsubscribe from a particular item of context must issue an unsubscribe request. When an unsubscribe request is made of a service the handle relating to the client application is removed from the service so that no further updates will be propagated.
- **Save**: In order to manage application state across sessions and devices, a safe termination of an application will result in a `save` command being issued. By requesting a `save` and supplying the session identifier and user credentials any state pertaining to a session can be stored by the local context server, which may in turn propagate these changes to the central context repository.
- **Restore**: Applications wishing to return to a previous session during initialisation can issue a `restore` command along with a session identifier and user credentials. Any relevant state can therefore be retrieved from either the local CSP or the remote context repository.

The adoption of a subscription mechanism affords context-aware applications with the ability to register for particular events and then move freely throughout a distributed environment and still have updates delivered in a timely manner. This is discussed in more detail in section 6.3.4 and summarised below in figure 5.9.

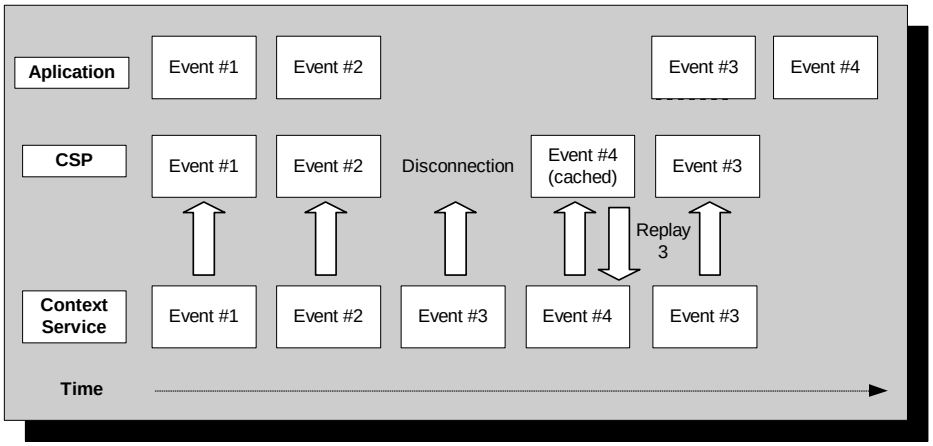

Figure 5.9 - The replaying of events following disconnected operation

5.3 Infrastructure in Use

5.3.1 Overview

This section illustrates how the architecture described in the previous section was used to author two simple prototype context-aware applications. The two prototypes described below were developed in parallel with the GUIDE II prototype (detailed in chapter six) in order to create a number of smaller application demonstrators within the domains of guide systems and office assistants (as described previously in table 5.1). In essence, they were designed to take several of the original GUIDE application features, namely the location-awareness and communications tool, and allow them to be further developed within a working laboratory environment independently of the rest of the system. In more detail, the use of location awareness in conjunction with a universal communications tool (i.e. a messaging application called Ubichat) provides a good foundation with which to experiment further with the CSP architecture using a real user base (i.e. members of the department). The first prototype application described, Ubichat (ubiquitous chat), allows users to send instant messages to users on their contact list irrespective of the sender or receiver location or the device. The second prototype application described, the digital In/Out board provides a simple awareness mechanism relating to the location of the department's Distributed Multimedia Research Group (DMRG) members.

5.3.2 The Ubichat Application

5.3.2.1 Introduction

The initial motivation for Ubichat was established after the success of the message service incorporated into GUIDE application. The GUIDE message service proved to be a very popular tool despite several limitations. These included the inability to communicate with users whilst they were experiencing periods of disconnection and the lack of feedback relating to the status of sent messages. During a user field trial a short message service (SMS) gateway was incorporated into the GUIDE system to provide a simple mechanism for allowing GUIDE instant-messages to be sent over the GSM network. This enabled users of the GUIDE system to communicate with users outside of the system. Ubichat provides a general messaging system and is able to send/receive messages of different formats to/from a range of devices. The application currently supports three message formats:

- **Email**: Messages may be via a POP mail server to the intended recipient.
- **Short message service**: Messages may be sent via the GSM network as a short text message.
- **Ubichat message**: Messages may be sent directly to the recipient's Ubichat client application.

When a user initialises their Ubichat application, they are presented with an interface that is similar to other instant messaging applications such as ICQ or MSN Messenger, shown in figure 5.10. Users are presented with their personal contact list (retrieved from the context service provider based on their username and password pair), for example, friends, work colleagues, etc. The contact list not only displays the names of other users but also their status or context. More specifically, the status of a user reflects their current location, device they are using and, finally, an indication regarding the validity of the context. This contextual information is used to provide users with an awareness of other users and also their device constraints. Furthermore, this awareness information is used by default to adapt the delivery mechanism adopted when sending messages to members of the contact list. For example, if the system detects that a user is located in their office, a message will be sent directly to

their Ubichat application; however, if the system detects that they are out of the office, a notification may be sent to the user's mobile phone.

5.3.2.2 Building the application

When Ubichat is launched, it registers with the context service provider and specifies the context of interest, in this example, *location* and *user* services are required. The user manager service is responsible for maintaining a structure of a user's current context, that is, their location and where possible their device type. Updates to user context are reflected at the interface level and take into consideration the user stipulated privacy options, that is, a user is only made aware of other users within their contact list, as shown in figure 5.10.

Figure 5.10 - Ubichat interface for both Palm and Windows platforms

The user manager service abstracts over the process of collecting the context relating to a user and their context. From the application developer's point of view, the user manager provides a number of simple operations that enable a user's location to be determined and hides the complexity of how the context is collated. In more detail, the Ubichat application is able to collect location updates from a variety of sources (see figure 5.11).

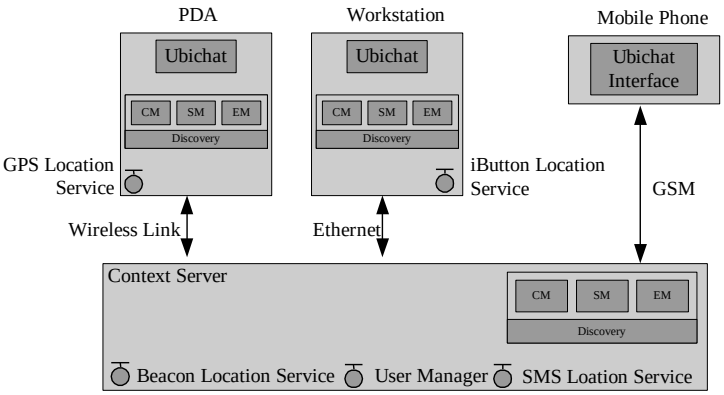

Figure 5.11 - Ubichat architectural components

Updates may be posted to the user manager from a number of sources, including:

- **GPS**: A user of the system utilising a GPS device for location updates may notify the user manager of their location updates.
- **iButton**: A number of Blue Dot receptors, or iButton readers, are located throughout the department building and an application monitors the state of the reader and waits for users to 'dock' their iButtons, as shown in figure 5.14.
- **Cellular Infrastructure**: The cellular based wireless infrastructure periodically transmits unique identifiers, or beacons, throughout the city, university campus and department offices and laboratories.
- **SMS**: Users may update their context manually by sending an SMS message to a pre-defined mobile phone number. This number belongs to a WaveCom GSM module which is permanently connected to a PC running Linux. A software component running on the device monitors the inbox and parses the SMS messages. SMS messages with the syntax shown in figure 5.12 are processed and location updates posted to the user manager.

```
CMD UPDATELOC USERNAME LOCATION PRIVACY
CMD UPDATELOC KEITH OFFICE ALL
CMD UPDATELOC NIGEL HOME GROUP DMRG
```

Figure 5.12 - The format of a sms message

The Ubichat application makes use of the contact list to determine which users to register an interest with. By registering with the user manager and supplying a list of usernames, the user manager can track sightings of members of the contact list on the

application's behalf. Any updates received by the user manager are propagated to the relevant client applications and their interface is updated accordingly. Figure 5.13 shows how a client application is able to receive location updates from any number of location services. It is also shown how, through the use of the central context repository (residing in the network), a client's session can be saved and restored on a separate device.

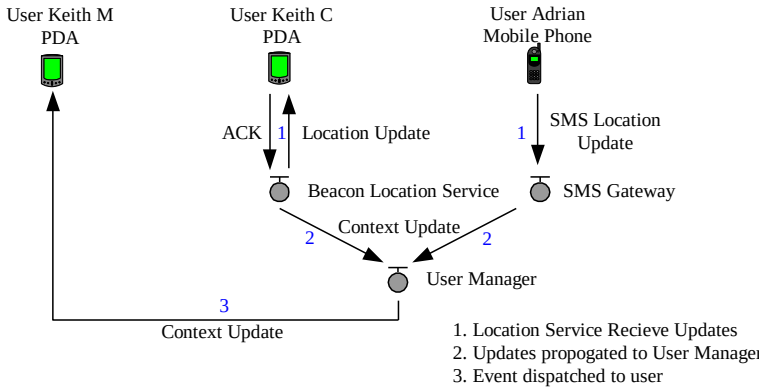

Figure 5.13 - Receiving a location update

To send a message, a user simply double clicks on the name of the person they wish to contact before entering the message content. The message is then delivered to the message service which determines how best to dispatch the message based on the recipient's context.

5.3.3 The Department's Public In/Out Board

5.3.3.1 Building the Application

The digital In/Out board application, as shown in figure 5.15, was designed to provide a simple awareness mechanism for staff and research students to quickly determine the availability and location of academic staff within the department.

Two versions of the In/Out application are available, the first is a simple Java client that runs locally on a PC with a Blue Dot receptor attached. The client monitors the state of the reader and waits for users to 'dock' their iButtons [iButton,00a], as shown in figure 5.14 below, before forwarding the iButton sightings to the user manager. The client also registers with the user manager and displays the in/out status of the

research group members contact list. Additional information displayed includes their current (or last known) location, a timestamp recording their last sighting, and a hypertext link to a log file which contains a record of all their previous sightings within the building.

Figure 5.14 - The In/Out board iButton reader

The second version of the application[1], provides a simple web based interface to the system which displays similar details.

Figure 5.15 - The web interface to the department's In/Out application

When the In/Out application is launched, it registers with the context service provider and specifies the context of interest, in this example, *location* and *user* services are required. The user manager service which forms part of this application is the same service described previously in relation to the Ubichat application. As a result, both applications executing on the same device will receive updates relating to a user's location context simultaneously. This property enables applications to share context

[1] Available internally (for security and privacy reasons) at http://athens.comp.lancs.ac.uk/inout

common among different executing processes. By default, the digital In/Out client registers with the user manager and is able to retrieve location updates for all users of the system. However, by specifying privacy constraints, the visibility of a particular user can be restricted to a specific group or specific users. Any updates received by the user manager are propagated to the relevant client applications and filtered before their interface is updated. Figure 5.16 shows the architectural components which relate to the digital In/Out application.

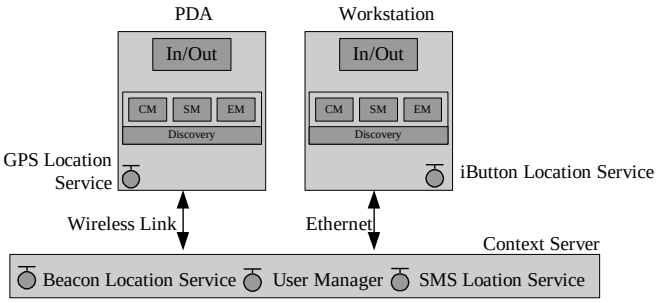

Figure 5.16 - The digital In/Out application architecture

5.3.4 Analysis

The primary advantage of the approach presented in this chapter, in relation to the above applications, is that the use of a context service enables applications to share context (state) both at run-time and between different application sessions executing on (perhaps) multiple devices. More specifically, since the context service provider is responsible for session management all interactions that take place are made persistent after the lifetime of the application (discussed further in chapter six). Should a user then initialise a new instance of a client application, say on an alternative device, then data relating to a previous session can be quickly restored and accessed (including contact list, history, etc). Moreover, since the context service provider acts as an intermediary for applications, different applications are able to access context or re-use the same context maintained centrally, i.e. maintain the same view of contacts, history, etc (see figure 5.17 below). This enables a user to work and move between devices during the course of a day and to maintain consistency irrespective of their device or application.

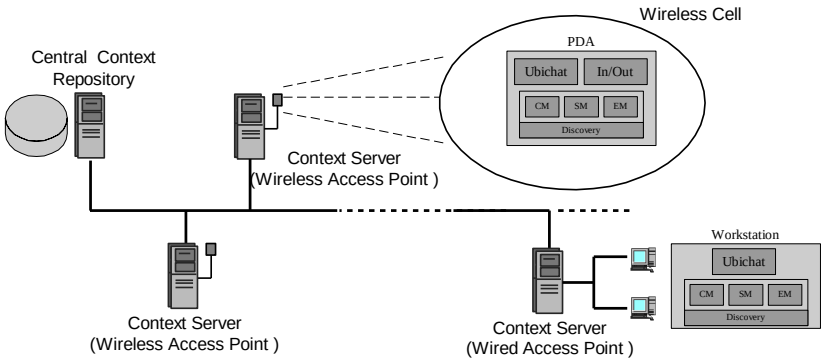

Figure 5.17 - Shared access to context

Furthermore, the implementation details which relate to specific services are hidden with access to the functionality obtained through standard interfaces. Since the use of context is controlled through the specification of context types, there is no inherent coupling between any particular application and a service offering the context of interest. As a result, should a new context service be added to the execution environment offering similar functionalities, the use of context specification and context discovery provides an application with the ability to adapt to changes in the environment. For example, when considering the applications detailed above, should the use of an alterative location technology be included into the computational environment (e.g. RF-id tags), then all that would be required by a system designer would be to create a service that acts as a wrapper for the underlying technology. Then, once instantiated, this executing service will become part of the context space and client applications can be notified of its presence and be able to make use of the service if required. A more in-depth analysis of the context service is described in the evaluation detailed in chapter six.

5.4 Summary

This chapter has described the design and implementation of a context service based architecture to provide developers of mobile context-aware applications with a useful mechanism for building applications designed to operate in mobile and potentially volatile environments. The key features of the context service based architecture are:

- The provision of a standard interface enabling clients to utilise a range of context services through a single interface;

- A mechanism for enabling users to stipulate their context constraints;
- Transparent session management features that support context-aware applications within mobile environments, that is, implicit support for user mobility including disconnected operation and roaming between multiple devices;
- The sharing of context amongst applications and application sessions;
- The development of context services that are re-usable across a range of applications, including inherent support for a variety of portable end-systems and platforms;
- An architecture capable of handling both synchronous and asynchronous modes of communication;
- Support for the dynamic discovery and utilisation of context services.

The main aims of the CSP are therefore to support mobile-users irrespective of their application or device and facilitate intrinsic support for ubiquitous context-services, user mobility and discovery within a highly dynamic environment.

The features provided by the context service were based on a set of requirements identified through an analysis of related work presented in chapter two and evaluation of the approach to mobile context-aware application design adopted as part of the Lancaster GUIDE project. The author believes that these features provide a solution to the problem of supporting context-aware applications designed for use in dynamic mobile environments. The following chapter evaluates the context service based architecture through a number of real scenarios and shows how, through re-engineering of the GUIDE prototype, the system can more readily adapt to its execution environment.

Chapter 6

Evaluation

6.1 Introduction and Methodology

The previous two chapters detailed the computational and engineering models relating to the Context Service Provider (CSP). This chapter evaluates the CSP and, more generally, the use of context services as a mechanism for supporting context-aware applications in mobile environments. The focus of the evaluation is entirely qualitative in that the chapter aims to validate the set of requirements for building context-aware applications and the effectiveness of the approach for experimenting within mobile environments. To achieve this, the evaluation takes the form of prototype implementation and experimentation to illustrate the flexibility provided through the use of context services.

A quantitative evaluation of the architecture detailed in this thesis is not provided since it is difficult to identify which specific items to evaluate or indeed what benefits are afforded by such an evaluation. First, it is difficult to specify a representative evaluation scenario that would provide typical values for the large number of contextual variables that may affect the overall system performance, such as the mobile devices and communications technologies being employed. More specifically, the GUIDE II prototypes and CSP make use of a variety of communications

technologies all of which share the university backbone network. As a result, the communications medium and available bandwidth is shared making it difficult to ascertain any potential benefits of this approach over an alternative approach within this environment. Furthermore, the lack of meaningful benchmarks that may act as reference values further complicates the interpretation of any quantitative measurements. Second, the prototype CSP implementation used in the evaluation has not been optimised for performance and was essentially created as a proof-of-concept implementation. Therefore, any performance measurements such as overall client latency, that is, the overall time required to discover, select and use a service after entering a new context, would be highly dependant on the discovery mechanism employed. Recall from chapter five that this was not a primary concern for the work detailed within this thesis and considered a valuable area for future work (see section 7.3.1). To some extent, the fact that the architecture is accessible from a range of mobile end-systems and has been deployed across a range of Linux and Windows based servers around both the city and campus network is of more relevance since this affords a basis for experimentation.

The evaluation therefore evolves as follows. First, section 6.2 will focus on the experiences gained from re-engineering the GUIDE application, referred to as GUIDE II, to make use of the common infrastructure detailed in chapters four and five. The overall architecture will be detailed first, including the major differences with respect to the original implementation. This will test the hypothesis relating to flexibility and demonstrate how an application (i.e. the GUIDE II prototype) is able to execute across a wide range of mobile end-systems whilst making use of the common infrastructure. The GUIDE II prototype will then be utilised to further evaluate the underlying infrastructure using a number of *real* scenarios that demonstrate the key features effectively (section 6.3). More specifically, it will be shown how the GUIDE II application benefits from the support provided by the context service based architecture in terms of application portability, flexibility and evolution. Furthermore, the architectural features enabling context-aware applications to be used in a dynamic mobile environments will be demonstrated. Following this, section 6.4 evaluates the CSP with respect to the requirements presented in chapter four, before, finally, a summary of the evaluation is described in section 6.5.

6.2 Re-engineering the GUIDE application: GUIDE II

6.2.1 General Approach

Many of the requirements for building the context-service based architecture were derived from a critique of the GUIDE prototype application. It is therefore instructive to re-engineer the GUIDE application to exploit the features in order to verify the initial requirements and to qualitatively evaluate its suitability within the mobile application domain. To facilitate operation with the underlying infrastructure the design of the GUIDE II application centred around the following modifications:

1. The design of application modules to support interaction with the underlying architecture;
2. The generalisation of core application services, such as the tour guide component, the message service and location service, enabling simultaneous access by multiple applications;
3. Extensions to the user interface to provide users with a mechanism for stipulating their context constraints and a suitable mechanism for providing feedback pertaining to the state of the operating environment, and;
4. The development of alternative user interfaces to facilitate access to the GUIDE II service infrastructure from a range of portable end-systems.

6.2.2 Application Design: General Architecture

A key objective for redesigning the GUIDE prototype application was to employ a more general approach and facilitate interaction with the CSP and context-services detailed in the previous chapters. Since the original implementation was specific to Lancaster and relatively difficult to extend, the GUIDE II application was designed to overcome these specific limitations and, by supporting incremental development of new functionality, include enhanced support for distributed environments. Initially, the functionality represented by each of the GUIDE constituent elements were isolated and redesigned to use the common infrastructure detailed in chapters four and five, the focus being to demonstrate access to the functionality from a range of mobile devices and not wholly the GUIDE unit presented in chapter three [Fujitsu,98]. To facilitate this the application's interface was separated from the core functionality

allowing GUIDE services to be rendered on multiple interfaces (i.e. physical devices). In essence, the GUIDE II application can be regarded as two primary components: the user interface and the agent (or proxy) component, which contains a significant proportion of the application's functionality. Since HTTP is used as the primary means of communication between system components, it is possible to execute the user interface and other components as separate processes without any adverse affects in performance. This separation of concerns allows multiple interfaces to be designed targeting the attributes afforded by specific mobile devices while, simultaneously, the core services remain device and platform agnostic. Figure 6.1 summarises the GUIDE II system components with respect to the underlying infrastructure and the following sections discuss the modifications in more detail.

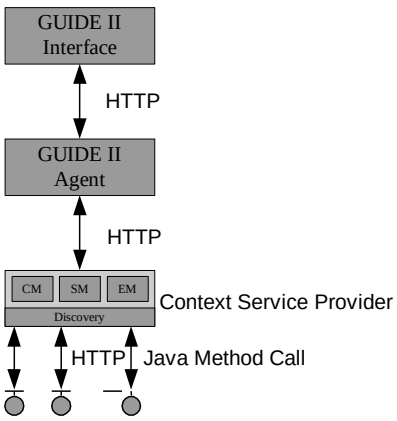

Figure 6.1 - A summary of the GUIDE II system architecture

6.2.3 Generalising the Core Application Services

The two major GUIDE components required to be modified were the location mechanism and the information model. The use of location beacons broadcast over the wireless communications channel provide the GUIDE application with positional information. This approach provides varying location accuracies depending upon the location of the GUIDE cell server, since the city's geography affected the propagation of the radio signal quite dramatically, as discussed in section 3.3.2.2. Furthermore, the GUIDE application, and therefore the user, remain unaware of these inconsistencies and the varying levels of connectivity afforded by the infrastructure. In essence, only a simple visual indicator representing either full or no coverage is supported at the user interface level, as shown in figure 6.2 [Cheverst,99a].

Figure 6.2 - Network connectivity at the user interface level

Thus, when designing the GUIDE II application, the location services were re-engineered with the ability to express the type of service they offered and the level of service (QoS) available, as described in section 5.2.4. For example, a location service is able to advertise to applications the accuracy it provides (e.g. 10m or 100m). Primed with this knowledge an application may determine and select the most suitable service by trading-off the various properties. With this in mind, the location services were re-engineered as generic context services as described in section 5.2.2.

With respect to the information model, the GUIDE system was tightly coupled to the Lancaster object model and was unable to load objects dynamically. Furthermore, the prototype required that all objects be stored locally and that they be processed upon application initialisation. This approach has several limitations: first, when modelling a relatively compact city such as Lancaster, the number of objects required is quite small[2] and may therefore reside locally with relative ease. However, when modelling a large city, such as New York, a more scalable approach is required since it may not be possible to store large data sets relating to an entire city locally. Second, it was discovered during an evaluation of the GUIDE prototype [Cheverst,00a] that a requirement to support virtual navigation exists [Benyon,97]. For example, information pertaining to a particular landmark in Lancaster may contain a reference (i.e. a hypertext link) to a landmark in another city. By following this link a user decides to visit the landmark or city virtually, and therefore, the relevant data model will need loading dynamically. Therefore, a single system capable of dynamically downloading any number of information models would enable a user to pre-plan a trip to a destination from home and then allow access to their tourist itinerary as they travel. Moreover, a tourist visiting several destinations will be able to obtain information dynamically on demand.

[2] Currently over 450 objects (location, navigation, neighbour objects) exist, which is enough to model the city centre of Lancaster, and requires approximately 675 K bytes of disk space.

6.2.4 User Interface Extensions

To provide users with a convenient mechanism for associating contextual constraints with their user preferences, a number of user interface enhancements were required. Furthermore, modifications were also required to ensure that feedback relating to the state of the environment was clearly represented.

By default, the GUIDE application requires the use of several context services including: a location service to provide positional information to users, a tour guide component to provide a model of the geographic environment and for navigation tasks, and a messaging service to support communication between GUIDE users. Since location acts as a valuable form of context the GUIDE II application enables users to stipulate several QoS constraints associated with their location. In essence, users are able to control several parameters in relation to making use of a location service: the frequency of updates required, the approximate battery consumption required and the positional accuracy required (as shown in figure 6.3 and appendix A). It should be noted that a trade off exists between positional accuracy and power consumption. More specifically, a user who requests a higher degree of positional accuracy is assumed to require a location mechanism such as a local GPS compass. In a situation where positional accuracy is of less importance an alternative technology, such as the beacon based approach detailed in chapter three, may be utilised since this is likely to reduce the battery consumption of the mobile device.

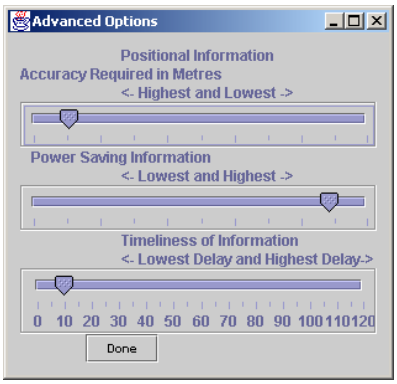

Figure 6.3 - User specification of context constraints in GUIDE II

With respect to the information model utilised by the GUIDE system, users may specify their tour guide constraints by stipulating both their user preferences and the city they are interested in exploring (see appendix A). As detailed in the previous

section, the original GUIDE prototype was tightly coupled to the Lancaster information model. Therefore, the GUIDE application was modified to interact with the underlying CSP in order to utilise the discovery features provided and to enable dynamic binding to available information models. During application initialisation, the GUIDE II application broadcasts a discovery packet specifying the context constraints, as shown in figure 6.4.

	Name (identifier)	**Condition**
Context Type	Type	Location
Context Constraints	Accuracy Delay Battery	(GREATER or EQUAL) 25 LESS 10 *

Context Type	Type	Geographic Model
Context Constraints	City	*

Figure 6.4 - GUIDE II discovery packets

The asterisks (*) shown in figure 6.4 represents a wildcard, denoting that no specific constraints have been specified and so the results of a search for all context services of this type will be returned. Based on the above discovery packets, the following user interface, shown in figure 6.5, is presented to a user of the GUIDE II application which identifies several geographic models available for use.

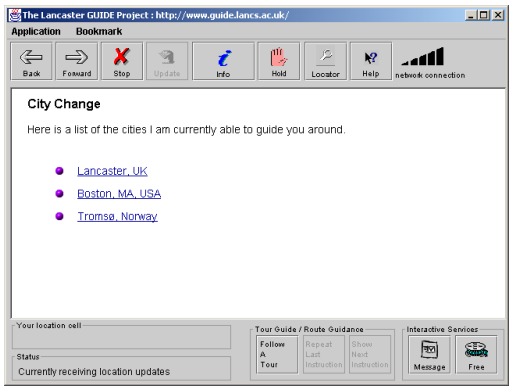

Figure 6.5 - GUIDE II geographic service selection

6.2.5 Developing for mobile devices

6.2.5.1 Overview

The GUIDE II application was designed to facilitate operation across a range of portable devices and, as introduced in section 6.2.2, the user interface can be designed

targeting specific mobile device characteristics. In essence, issues such as display characteristics and available screen real estate were pertinent to the development of the user interface. For the purposes of the evaluating the CSP architecture, the GUIDE II prototype application was designed for each of the following devices (summarised in figure 6.6), since they are widely available and reflect the current state-of-the-art at the time of writing:

- **Pen Tablet**: A GUIDE II application was developed targeting the original GUIDE unit, the Fujitsu TeamPad 7600 [Fujitsu,98].
- **PDA**: A GUIDE II application targeting the Compaq iPAQ Pocket PC PDA was developed which enables context-aware information to be accessed via the Pocket Internet Explorer web browser [Sun,00d].
- **Mobile Phone**: A GUIDE II application targeting the Wireless Application Protocol (WAP) [WAP,00] was developed and tested using both a Siemens C35i WAP enabled mobile phone and a WAP Emulator. In this version, access to context-aware information was obtained remotely with no processing carried out locally apart from the rendering of WML decks and cards by the WAP browser.

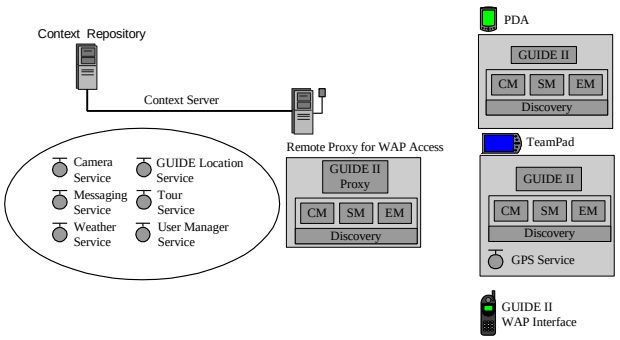

Figure 6.6 - A summary of the GUIDE II prototypes

6.2.5.2 Accessing The GUIDE Infrastructure using a PDA

The PDA interface is based around the Compaq iPAQ Pocket PC [Compaq,01] and provides a standard web interface to the GUIDE II infrastructure. The Pocket Internet Explorer web browser is configured to access a HTTP proxy which is executing locally [Sun,00d]. This local proxy (see section 3.5.4) has been modified to register with the CSP and, in terms of application functionality, operates in exactly the same

way as detailed in chapter three. To enable a similar 'look and feel' to the existing GUIDE application, the GUIDE filter (see section 3.5.5) has been modified to dynamically adjust the hypertext content based on the context of the mobile device. The filter dynamically inserts a header before displaying a hypertext page which includes a user's location context and a series of buttons relating to the main tourist services available. The location context describes either their current or last known position as shown in green or red respectively in figure 6.7.

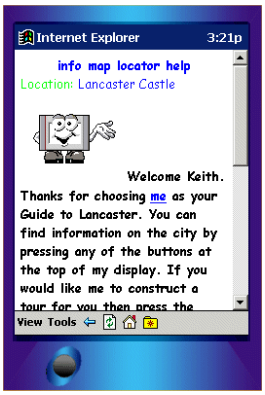

Figure 6.7 - The GUIDE II interface for a Pocket PC PDA

6.2.5.3 Accessing the GUIDE Infrastructure using WAP

To access the context-aware services offered by the GUIDE II infrastructure users of an ultra-mobile device, such as a WAP enabled mobile phone, are able to connect to the remote GUIDE proxy component. This proxy makes use of an additional filtering service to translate tourist hypertext pages into WML decks and cards designed for display by a WAP browser. The GUIDE WAP interface supports functionality including the ability for new users to login and create a profile, in addition to allowing existing users of the system to login, using their username and password and retrieve an existing profile. All user requests are processed by the proxy component which stores and retrieves their personal information from the context service provider. By allowing access to the user profile in this manner context-sensitive content can be delivered to a mobile user using WAP.

Figure 6.8 - The GUIDE II WAP interface

6.2.6 Summary

This section has detailed the design of the GUIDE II prototypes based on a series of modifications to the original GUIDE implementation presented in chapter three. More specifically, two additional prototypes, based on a PDA and mobile phone, were introduced which facilitate access to the GUIDE tourist services and CSP infrastructure via a range of portable end-systems. This provides a suitable mechanism for evaluating the features of the CSP architecture since it can be shown it is able to operate across a wide range of devices.

The experiences gained through this process has helped test and prove the hypothesis that the use of context services simplifies the development of mobile context-aware applications. In particular, the application development process was greatly simplified once a suitable underlying infrastructure consisting of the CSP and (GUIDE) context services were created. Indeed, the WAP and PDA versions of the GUIDE prototype was developed significantly more rapidly than the original implementation. More importantly, modifications to the GUIDE context services requires little or, in most cases, no further modification to the prototype applications since no hard-wired reliance exists between the application functionality and the specific service implementations.

The current prototype CSP implementation has been developed using JDK 1.1.8 and the PersonalJava 1.1.1 specification. This configuration ensures that the architecture can be deployed within a range of devices operating under a variety of different platforms. More specifically, the CSP implementation detailed in this thesis has been developed and tested using the Windows 95, Windows NT, Windows 2000 and Windows Pocket PC operating systems. Furthermore, these have been executed on a range of computational devices including a Compaq iPAQ Pocket PC, Dell Latitude

CS Notebook and a range of Pentium Pro, Pentium II and Pentium III personal computers. Furthermore, the range of services and applications detailed herein have also targeted the above platforms and devices.

6.3 Test Scenarios

The following section presents a further qualitative evaluation of the programming, service-deployment and mobility support aspects of the CSP. The bulk of this section is devoted to a GUIDE II case study and investigates through a number of *real* scenarios the extent to which the architecture provides the necessary support for context-aware applications to execute in a dynamic environment.

For the benefit of the reader, recall from section 4.3.1 the primary design goals:

- to allow a user to work and move between devices during the course of a day;
- to maintain a consistent view of the state of their environment irrespective of device or application being used;
- to include support for application adaptation and state management in relation to a changing execution environment involving the discovery and utilisation of new services.

These form the motivation for the following usage scenarios and will each address a key consideration in supporting a user within a dynamically changing execution environment in which periods of network disconnection are prominent, the availability of services fluctuate rapidly, and the device utilised to provide access to services changes over time.

6.3.1 Scenario One: Supporting Applications in Changing Environments

6.3.1.1 Introduction and Motivation

This scenario simply considers the introduction of a context service into an already executing system and the mechanisms enabling client applications to utilise the service dynamically. This allows a client application to remain usable despite frequent changes to the environment. The motivation for this test stems from the

GUIDE prototype application, where the inability for users to select services based on their current context given the existence of a number of alternatives was seen as a major drawback. More specifically, the wireless network utilised to provide users with location information provides relatively coarse-grain accuracy in terms of cells; that is, users are only able to determine which geographic region of the city they are located in. Therefore the system is unable to inform them that they are located near a specific landmark, such as a particular shop.

By supplementing the wireless infrastructure with a variety of more accurate location technologies, such as Active Badges, Bluetooth devices or RF-id tags, the GUIDE system is able provide users with more accurate location information. In a situation where alternative location technologies are made available, the GUIDE implementation is unable to dynamically make use of them even though the alternatives may poses more attractive properties (i.e. accuracies, power constraints). Within such a diverse environment, it may be valuable for a user to select services based on their current context. For example, following a guided tour of a city may require fine grained accuracy whereas accuracy may be less important to a user simply browsing information whilst having coffee.

6.3.1.2 Implementation

The overall scenario, shown in figure 6.9, considers the introduction of a new location service into an already executing application session. In the current implementation a user of the GUIDE II prototype application specifies their location constraints initially during startup, and these are stored as part of the user profile (see appendix A).

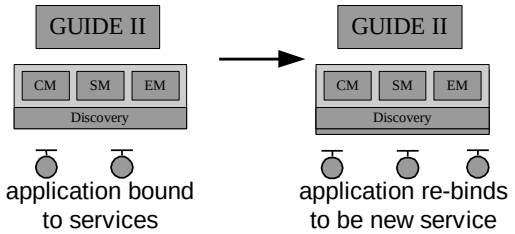

Figure 6.9 - Introducing a new context service

The GUIDE II application registers with the CSP and passes the user profile and context constraints to the context manager. The context manager extracts the service

types and constraints, as shown in figure 6.10 and stores them locally in a context registry. A key feature of the context registry is that the context services are classified in relation to their context types, representing a snapshot of an application's context requirements.

```
// Receive the data packet
services = recvdPacket.getBytes.getServices();
// Get the context type and user constraints
type = services.getType();
constraints = services.getContraints();
```

Figure 6.10 - Processing a context discovery packet

In addition, the CSP listens for service announcements dispatched by entities within the environment and maintains a context state table relating to the context types currently available. This operates as a collection of contextual sources so that the context manager is able to create bindings between applications and services based on the stipulated requirements.

Context Discovery Packet				
Type	**Description**	**Host:Port**	**Interface Details**	**Expires**
Location	GPS	localhost:6000	http://localhost/gps.xml	1000
Location	Nibble	localhost:7000	http://localhost/nibble.xml	30
Camera	QuickCam	194.80.35.25:2000	http://194.80.35.25/cam.xml	300

Table 6.1 - Context service state table

The services available are enumerated and matched against the services requested by a client application. A service match based purely on context type results in a further query to inspect and evaluate the properties of the service. In more detail, during this stage the context constraints specified by a user determine the suitability of available services, for example, table 6.1 (shown above) identities two location services and a camera service available for use by a client application. The method of evaluating a service is similar to the way in which the GUIDE prototype evaluates the suitability of a particular tour. More specifically, the constraints and parameters specified are compared to the properties offered by a prospective service. Successful attribute matches result in a positive increase in the weighing associated with that service. The service with the best overall weighting is selected and returned to the client application since this is considered the most appropriate service available at that time. The following pseudo-code summarises the service selection process.

```
for all discovered services stored in the context state table
{
       if (user specified context type equals context service type)
       {
              get user-profile(constraints)
              evaluate the service(constraints)
              store result in a temp Vector
       }
       else
       {
              no match found
       }

       enumerate vector and select service with highest weighting
       return service to user
}
```

Figure 6.11 - Context service evaluation

From here, by default, the CSP will aim to select a service automatically based on the user requirements, thus avoiding the need to interrupt a user on a regular basis since it is assumed that the environment is unreliable and that frequent service disconnections are common. However, in the event that several services are perceived to be relevant and the CSP is unable to select a service automatically, a summary containing the interface descriptions are provided to the user. A user is therefore able to select the required service manually by following a hypertext link (shown earlier in figure 6.5).

The selection of a context service, either manually or automatically, results in the CSP returning to the application the details pertaining to the communications channel, i.e. hostname and port number. In the event that a more suitable service (based on the user requirements) becomes available, the CSP is able to dynamically adapt to using the new service. In operation, all the handover actions are triggered by the receipt of a service announcement packet from an advertising context service. The service is registered with the CSP context manager and, if deemed more appropriate, the CSP removes the registration from its predecessor and subscribes to the new service. Although users are notified of service changes at the interface level, any modification to the communications channel remains transparent, as shown in figure 6.12 below.

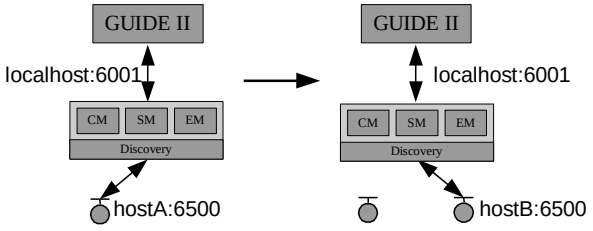

Figure 6.12 - Transparent context service re-binding

6.3.1.3 Analysis

This demonstration highlights one of the features of the CSP, mainly, its ability to dynamically reconfigure the services offered to client applications based on a user's stipulated context. In summary, the mechanisms provided by the context service based architecture to support the automatic configuration of context services includes:

- **Context Specification**: The use of *standard* service templates to describe services allows client applications to be designed generically in terms of context types, such as location, people, activity, weather, etc;
- **Context Constraints**: The use of context constraints governs the service selection process based on the current configuration, and;
- **Context Discovery**: The use of a simple context discovery mechanism enables applications to make run time bindings to context services.

As a result, the use of context is more explicit and client applications can be designed with context types and not specific hard-wired services in mind. Furthermore, adaptation based on user stipulated requirements provides a mechanism in which users are able to control, more readily, the constraints under which adaptation should take place.

6.3.2 Scenario Two: Maintaining a consistent environmental representation

6.3.2.1 Introduction and Motivation

This scenario considers a situation in which a user of the GUIDE II application begins an application session using a mobile end-system and then, during the session, switches to an alternative end-system affording different device constraints. This scenario will therefore demonstrate how the CSP is able to manage an application's context and state information relating to an active session. More specifically, it affords mobile users with the ability to roam between devices during the course of a day and facilitates access to their personal context irrespective of their device constraints. These operating conditions provide a sound basis for evaluating the architecture's effectiveness for maintaining a user's view of the environment despite

fluctuations and overcomes a limitation of the original GUIDE application which, despite maintaining state relating to a range of environmental contexts including a user's profile, did not persistently store or allow this state to be retrieved trivially.

6.3.2.2 Implementation

The test configuration is summarised in Figure 6.13 and depicts two instances of the GUIDE II application executing first on a GUIDE unit and second on a Compaq iPAQ Pocket PC.

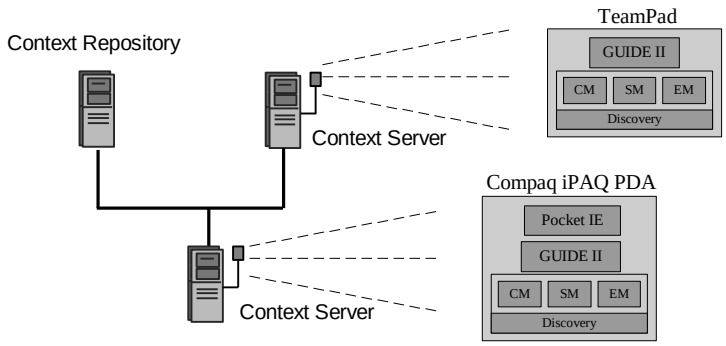

Figure 6.13 - Application configuration

The initial application session is executed on the GUIDE unit and all the initial session state pertaining to the active session is stored on the end-system locally in the CSP. When a user terminates the GUIDE II application the `save()` method is invoked in the local CSP. This generates a globally unique identifier which is returned to the application and saved locally (i.e. in the current implementation, the unique identifier is saved to the local file system). This identifier, along with the username and password, is used to identify the session uniquely. Once a unique identifier has been created, the session state is forwarded to the central context repository via the local context server.

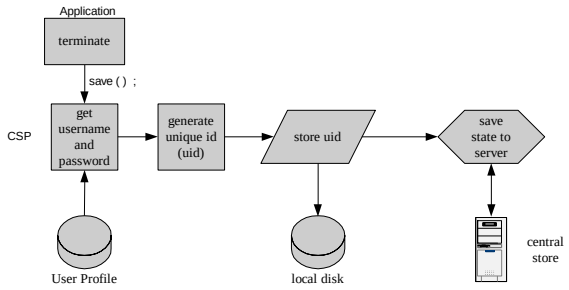

Figure 6.14 - Saving application state

A second instance of the GUIDE II application initialised on the same end-system will be able to gain fast access to the previous session state by issuing a restore command, which retrieves the session state from the local CSP. Should a second instance of the GUIDE II application be initialised on a different end-system, such as an iPAQ Pocket PC, then the `restore()` command processed by the local CSP will cause an exception to be thrown since no state exists locally. Following this an explicit request (containing the identifier) is forwarded to the central context repository along with a user's username and password pair, since the identifier will not be recognised locally. The central context server will then validate the supplied credentials with those stored in the repository. A successful match will result in the state being returned to the local CSP. The session can then be accessed using the new mobile device with the session continuing as normal with any previous state restored. However, an unsuccessful match will result in the GUIDE II application requesting that a user create a new session, since it is assumed that no previous state exists and therefore the user must be new to the system. Figure 6.15 provides a summary of this process.

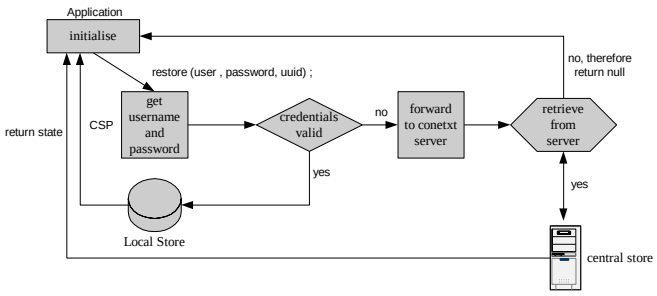

Figure 6.15 - Restoring application state

6.3.2.3 Analysis

The roaming of application sessions between mobile devices was not supported within the original GUIDE implementation, since application state was maintained locally and unavailable to other instances. However, this demonstration has shown that, through the use of several simple method calls, namely `save()` and `restore()` application state can be readily made available to multiple application sessions while maintaining user privacy. In the current implementation, the security measures are through the use of a username and password pair, although in the next chapter (see section 7.3.4) an extension of this approach is detailed, which is currently work-in-progress as part of the GUIDE II project [Friday,01b]. The GUIDE II project aims to provide secure and trusted access to distributed services via a wireless (802.11b) infrastructure. The overall approach is based on a modified Mobile IPv6 protocol stack that uses packet marking and network level packet filtering at the edge of the wired network to achieve this objective.

6.3.3 Scenario Three: Supporting shared access to context

6.3.3.1 Introduction and Motivation

This scenario investigates how the use of the CSP supports the sharing of context between system components (i.e. users) which can be used to support cooperation between geographically distributed users in a mobile environment within the GUIDE system. This encompasses a number of key issues including: supporting the sharing of location context (requirement R14), maintaining user privacy (requirement R4), presentation of location context (requirement R12) and managing the storage of positioning information (requirement R7). For the purposes of this scenario, consider the following situation [Cheverst,00d]:

"John, a visitor to a new city, is looking for a café to visit and uses his personal GUIDE to assist him. John is shown a series of relevant web-pages containing a number of cafés ordered based on their proximity and John's preference for vegetarian food. However, John is still undecided and reads a list of comments left by other visitors for the various cafés. Furthermore, John is shown that another GUIDE user, Mary, is currently located in one of the cafés listed. John would like to have an

unbiased opinion on whether the café is currently quiet and so decides to send a short message to Mary enquiring about the café."

Although supporting this kind of cooperation is relevant to the research fields of CSCW and groupware, to date, little research has investigated the ways in which mobile context-aware systems can be used to support cooperation and interaction between users. Moreover, there has been little research into the development of context-aware systems that enable mobile users to receive an awareness of other users whose location in the physical world corresponds to those places being explored in the virtual world.

6.3.3.2 Implementation

To facilitate the sharing of location information between system entities the user profile component was modified to enable users to explicitly state their privacy constraints (see appendix A). More specifically, a user is able to stipulate their privacy constraints relating to the context types of interest when they first initialise the GUIDE II prototype application. This enables context pertaining to a user's location to be queried by other system entities, providing the access privileges are satisfied.

In order to provide accurate place awareness relating to a given visitor, the system needs to place the visitor at specific locations, e.g. the City Castle or the Folly Café. However, the location information provided by original beacon based location mechanism was not able to provide this granularity of information. In order to achieve this greater level of accuracy a number of alternative techniques have been incorporated into the system including, the use of a simple translation service developed to convert GPS co-ordinates into GUIDE location identifiers [Byun,01]. In addition, a prototype service based on the Nibble Location service [Castro,01] (detailed below in section 6.3.5) was also incorporated to provide a means of accurate positional information.

User Privacy

A key issued raised by this scenario is that of user privacy. Previous experiments involving location tracking systems, such as the active badge system [Want,92], have

encountered difficulties relating to the unwillingness of users to allow their location to be known to the system. In the original GUIDE prototype implementation, no state information regarding the location of clients was made available to third parties and therefore GUIDE users were given no reason to feel that their location was being explicitly tracked. Indeed, during an evaluation of the GUIDE implementation no concerns pertaining to privacy of location were raised [Cheverst,00c]. However, visitors may feel concerned if they believe that personal context, such as their location, is available to others and for this reason, mechanisms such as user stipulated context constraints (Appendix A) enable users to control their anonymity. Figure 6.16 below shows the user interface presented to a user of the GUIDE II application. This screen forms a part of a larger profile creation wizard that allows a user to stipulate what context is to be shared and whether or not they wish to remain anonymous.

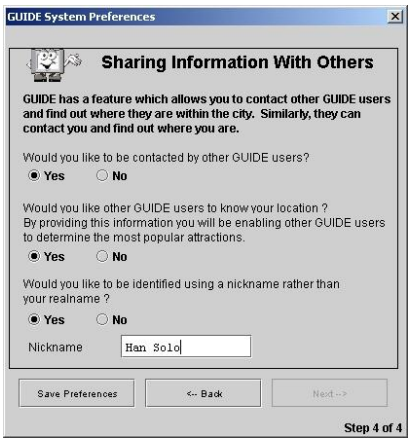

Figure 6.16 - User selects to remain anonymous to other GUIDE users

Managing User Location

The original GUIDE implementation which stored user location only locally enabled fast access to this information, even during periods of network disconnection. However, the longer this period of disconnection the greater the likelihood of the information becoming out of date or stale. Therefore a group of users all may have conflicting information pertaining to each other [Cheverst,99b]. Therefore, by creating a context service, such as the user manager, and storing this information centrally, location information will be consistent throughout the system and the management of privacy through access control relatively straightforward. One possible drawback with this approach is that disconnected visitors cannot access

location information, but given the dynamic nature of this information this is not expected to be a significant problem.

The mechanism for controlling service scope was based upon the notion of users, groups of users, and everyone (i.e. public access) and not a geographically controlled, (i.e. physical space) based mechanism since the notion of sharing files and folders between users and groups is a well understood desktop metaphor. The rational for this choice was to ensure that service discovery process was not limited to purely proximity based discovery and that it should facilitate the discovery of context services that may not be physically close to a user (i.e. within close proximity). This mechanism enables services to be universally visible or remain private to particular user. For example, a user may create a calendar service and limit access to this service by specifying that only the creator and other (named) work colleagues be able to access the properties.

User Interface

To represent a user's location, the GUIDE II application makes use of hypertext links within tourist pages. These links facilitate a simple mechanism in which to read or create new comments associated with a given attraction or user. Furthermore, by following a link representing a user's name, the process of sending a message to that user is initiated. Figure 6.17 shows how these links appear on a tourist page.

Figure 6.17 - GUIDE II interface presenting location awareness to the visitor

In the example shown above, when a GUIDE user clicks on the name 'Keith' the send message dialogue box will appear with the recipient's name automatically completed. Alternatively, a visitor may click on the 'collaborate' button at any time in order to

discover the location of all users around the city (given the appropriate permissions) and be able to read their comments, or publish their own comment relating to their location. Discovering another tourist relies on the use of the COLLABORATE GUIDE tag, which in turn invokes a search request for type `USER`. To summarise, the COLLABORATE GUIDE tag invokes a search request. This is then processed by the context service that maintains location information for mobile users (i.e. the user manager service). The context service retrieves the context for the specified landmark (i.e. it retrieves the user list for the landmark of interest). This context is then filtered according to the specified access privileges and if the user making the request is permitted to view the context of the individuals.

6.3.3.3 Analysis

This scenario demonstrated a novel approach for providing GUIDE users who are exploring an attraction virtually with an awareness of visitors that are physically located at the corresponding attraction. This extension to the GUIDE prototype demonstrates how the use of user stipulated context constraints enables an increased level of awareness between mobile users to be supported. Furthermore, this example displays the usefulness of the approach supported by the CSP architecture for enabling collaborative environments to be created more readily.

In addition, further benefits can be demonstrated when considering the use of multiple applications accessing the common CSP infrastructure, for example, GUIDE II and the digital In/Out board described in chapter five. Here, the In/Out client automatically updates the public In/Out web board once a location update is received on a user's mobile device. If a location update is not received from the infrastructure for N^3 seconds the In/Out clients updates the public board to indicate that the current context is temporarily unknown. By registering with the CSP the context type of interest (i.e. location service) both applications automatically have access to the context-sensing infrastructure. Furthermore, the layer of abstraction (indirection) between applications that use context and system components that provide context, enables the CSP to function as a context portal, that is, an access point responsible for

[3] In the current implementation, N is set to 120 seconds (i.e. 2 minutes).

managing context pertaining to both applications. One advantage of this approach is that several executing applications accessing the *context space* are consistent. For example, given the above scenario involving both the GUIDE II and In/Out applications, a consistent view of the operating environment is ensured since any location update received by the CSP is propagated to each individual application.

Finally, consider again the Ubichat application presented in chapter five for sending and receiving instant messages. Through the explicit registration process supported by the CSP, the Ubichat application is able to inform system entities of its device constraints (i.e. share context). This enables remote users to determine their colleague's status and, more importantly, their operating constraints. This clearly has implications in relation to user expectation and usage patterns when you consider a situation in which a user sat at their workstation is communicating with a colleague using a mobile phone. Clearly, the cost (in terms of the communication constraints in addition to cost) of sending a message from a mobile phone may affect the type of communications taking place between the two parties. More specifically, the user at their terminal is aware of the constraints of their mobile colleague; their expectation in terms of speed of response, length of reply may be much lower.

6.3.4 Scenario Four: Supporting disconnected operation

6.3.4.1 Introduction and Motivation

This scenario considers a situation in which a user of the GUIDE II application experiences a period of disconnection from the network. During the disconnected period, a contextual source disseminates a series of asynchronous events destined for several mobile users. In addition, a user also tries to make an explicit request for a resource (i.e. synchronous communication is attempted). This scenario therefore explores how the CSP is able to provide support for disconnected operation.

The motivation for this scenario arises from the GUIDE prototype described in chapter three. Despite the use of a broadcast approach to data dissemination and the use of distributed caches to support disconnected operation, the GUIDE application provides no guarantees relating to the reliability of the information model and it is possible for an application to be using stale information. Furthermore, explicit

requests for resources made during periods of disconnection from the network are unable to be satisfied and any contextual updates to the information model that occur during disconnected operation are not seen at the application level. This scenario will demonstrate how the session management features provided by the CSP aid applications operating in unreliable environments. More specifically, it describes the features provide by the common infrastructure to retrieve context despite fluctuations in the environment so that application developers do not need to handle this on a per-application basis.

6.3.4.2 Implementation

The test configuration is based around the GUIDE II application detailed in section 6.2. Two experiments will be detailed. The first demonstrates how the CSP supports disconnected operation when client applications use the synchronous (i.e. request/response) style of interaction with context services. Second, the asynchronous (i.e. publish/subscribe) means of communication will be demonstrated.

Synchronous Communications

During disconnected operation explicit requests for resources rely heavily on the state manager since this component acts as a local cache. An explicit request for a data item not resident within the local cache results in a cache miss. Since a cache miss cannot be serviced or masked from the user, the requested item is stored in a cache miss register (modelled as a Vector in Java) and the request initially appears to fail at the application or user level, as shown in figure 6.18.

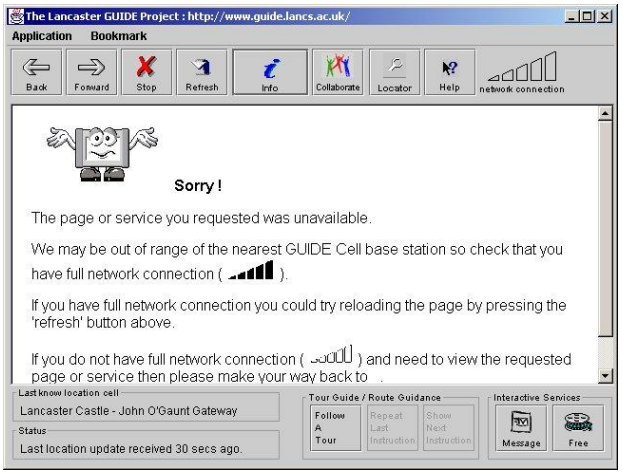

Figure 6.18 - Representing a cache miss at the application level

However, the state manager periodically attempts to establish a communications channel with a remote resource and endeavours to retrieve all requested resources once a network connection is re-established. Although this feature appears transparent to the user, they are able to control how and when the results of a cache miss are delivered. More specifically, the user is able to specify (during the user profile creation process described in Appendix A) one of three available options that allows the delivery of information to be tailored [Cheverst,01d], including:

- **Push**: A user request for data push will ensure that any item will be forwarded to a client application immediately once available.
- **Notify**: The notify option creates a level of indirection between a user and a new event or resource becoming available. In this mode of operation, an event, perhaps triggered by the availability of a resource, is forwarded to the client application and the user is then able to decide whether or not they wish to view the update at that time.
- **Cache**: This option will result in no visible update at the user interface level but will simply store locally (i.e. cache) the updated information, perhaps for later retrieval.

The management of the local cache is further aided through the use of context types, timestamps and expiry information. These attributes are used to define resource mutability in order to aid cache management. More precisely, the mutability of a resource can be used in the cache replacement policy, for example, a tourist hypertext

page of information representing a café may have a short expiry time. This may be used to indicate that there is little need to cache this particular data item since it is highly likely that the content changes rapidly (perhaps due to changes in the menu during the course of a day). However, hypertext resources such as history related pages for city landmarks or geographic maps of an area of the city may be cached since it is unlikely that these resources will have their content updated on a regular basis and may therefore reside within the local cache.

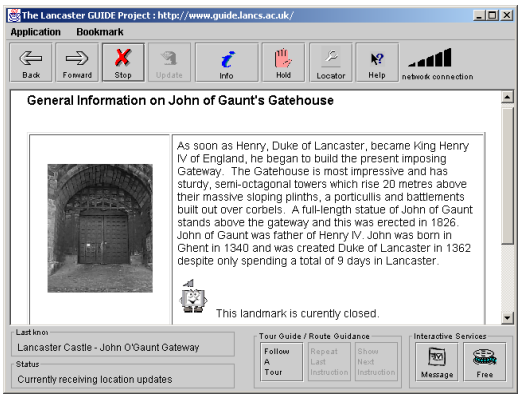

Figure 6.19 - Pushing an event to a user of the GUIDE II application

Asynchronous Communications

To support asynchronous forms of communication an application registers with the CSP in order to subscribe for events (as described in section 5.2.1.2). For the purposes of this example, assume that a service is available that announces the availability of tourist attractions, such as Lancaster Castle. While this is Lancaster's most popular tourist attraction, it is also home to a court room. Therefore, this particular landmark has a number of different tours available to tourists depending upon whether or not the courts are in session that day. A tourist interested in visiting Lancaster Castle may choose to have notifications relating to the state of the attraction delivered to their GUIDE application.

Assuming that a user is interested in receiving context events for Lancaster Castle, then should disconnection from the network occur the client application will be unable to receive context updates. However, the CSP provides two mechanisms for enabling context-aware applications to determine if an event was disseminated during a disconnected period.

The first mechanism is triggered autonomously whenever the CSP detects that network connectivity has been re-established. Once connectivity is detected, the generic `get(String name)` method of the `BaseContextService` class may be invoked to determine the current event identifier (i.e. sequence number) for that context service. Invoking this generic method requires the specification of the context service attribute that is to be queried. In this instance a call to `get("eventId");` will result in the context service returning the current state of that attribute. The result can then be evaluated against the current value for the event identifier stored locally in the CSP. If the identifiers are equal, then no events have been missed during the disconnected period. However, should a lower event identifier exist locally, then CSP may invoke the `replay(Context);` method in order to request the re-transmission of those events dispatched during the disconnected period.

Alternatively, should the CSP receive an event before it detects network connectivity (since the CSP checks periodically for network connectivity) then the incoming event is processed and the event identifier determined. If the event identifier does not directly succeed the current event id stored locally, then it is assumed that an event (or events) have been missed. In this instance, the `replay(Context)` method may be invoked and the current event is stored temporaily and processed once the previous events have been processed.

6.3.4.3 Analysis

This scenario has demonstrated that the use of the CSP provides applications with a mechanism for supporting disconnected operation when considering both synchronous and asynchronous forms of communication. Synchronous communication is supported using context-based cache management, that is, the caching of data items based on their context type, expiry and timestamp context to represent their resource mutability.

Asynchronous communication is supported by making additional use of sequence numbers (or event identifiers) as a valuable form of context. The combination of these attributes provides the CSP with enough information to determine if any context events have been disseminated during a period of network disconnection and to invoke the replaying of events.

One limitation of the current implementation is that the delivery of context events is controlled on a per application basis. More precisely, a user is able to control, using the push, notify and cache attributes, when context events are to be delivered; however, the chosen attribute is applied to all events received by that application. A more appropriate approach may allow individual applications to specify multiple constraints relating to different context types. For example, location updates may be required to be pushed and delivered immediately but other types of context events may be less important. Allowing a user to control how and when adaptation should take place based on complex policies is discussed further as an area of future work in chapter seven (see section 7.3.3). In addition, it is assumed that events replayed after connectivity has been re-established do not affect any other aspect of the context-aware system. More specifically, all events are assumed to be independent and do not affect other entities within the system. Investigating how event playback affects other entities within the system is also considered an area of future work.

6.3.5 Scenario Five: Supporting Application Extensibility and Portability

6.3.5.1 Introduction and Motivation

The above scenarios provide a mechanism for evaluating the key features of the architecture detailed in chapters four and five. Moreover, they consider the execution environment from an application's perspective and investigate how to manage the availability of new resources within a distributed environment. Of equal significance is exploring the application developer's perspective and to explore the flexibility of the architecture in terms of deploying newly developed services.

This issue of system flexibility will be addressed by demonstrating application extensibility, portability and evolution in turn. This has been partly addressed in 6.2 which demonstrated the architecture in use on a variety of platforms, physical environments, and portable devices (i.e. pen tablet, PDA, mobile phones). Extensibility and evolution will be addressed simultaneously by describing the steps involved for application developers to create service templates and generate context services. More specifically, this will involve the integration of the Nibble location system [Castro,01] in order to provide the GUIDE application with an alternative location technology.

6.3.5.2 Implementation

The Nibble location system is an indoor location system for mobile devices equipped with a wireless network card developed in the Multimedia Systems Laboratory at the Department of Computer Science, UCLA. The current implementation is limited to supporting Lucent Orinoco cards running under Microsoft Windows. The Nibble location service uses Bayesian networks to infer the location of a wireless client from signal quality measures. Furthermore, a device running Nibble can "remember" a location simply by associating a name with that location. Nibble utilises the signal quality received from access points detected at each location and incrementally builds a Bayesian network which can be used to calculate the most likely location for a signal quality "signature". The initial implementation has been used to provide location accuracy of approximately 10 feet apart (i.e. office room granularity), although the performance is highly dependant on factors such as building topology, number of access points, path effects and noise.

Although the current implementation is very limited, it serves a useful purpose in demonstrating the relative ease with which a context service can be created and incorporated into an already executing system. The limited Nibble API allows instances to be created and terminated through the use of `on()` and `off()` method calls respectively. Once executing, method calls to `getLocation()` or `getDistribution()` may be used to return the most likely location or the distribution strings, as shown in figure 6.20.

```
// Create an instance of the Nibble Location Service
Nibble thisNibble = new Nibble();
thisNibble.on();

// Get the location and distribution
String location = thisNibble.getLocationName();
String distribution = thisNibble.getDistribution();

// Stop the Nibble location service
nibble.off();
```

Figure 6.20 - Creating and using an instance of the Nibble location service

Let us assume that a location service offering similar functionality to that of the original GUIDE beaconing service is to be created. In essence, this service is to provide a string representing a location to client applications obtained by querying the Nibble location described above. The location service API provides applications with

the method `getLocation()` to obtain a positional update. Other information represented by the interface includes the accuracy provided by the service and the frequency of updates supported by the service. To create a location context service an XML representation describing the service interface is first created, as shown in figure 6.21.

```
<?xml version="1.0"?>
<context-service>
    <serviceType>Location</serviceType>
    <serviceId>Nibble</serviceType>
    <serviceChannel>
       <host>localhost</host>
       <port>6000</port>
    </serviceChannel>
    <eventChannel>
       <host>localhost</host>
       <port>6100</port>
    </eventChannel>
       <serviceStateTable>
           <stateVariable>
              <name>Location</name>
              <dataType>string</dataType>
           </stateVariable>
           <stateVariable>
              <name>Frequency</name>
              <dataType>integer</dataType>
           </stateVariable>
       </serviceStateTable>
       <actionList>
           <action>
              <name>getLocation</name>
              <paramterList>
                     <paramter>null</paramter>
                     <paramterType>null</paramterType>
              </paramterList>
              <returnType>String</returnType>
           </action>
           <action>
              <name>getFrequency</name>
              <paramterList>
                     <paramter>null</paramter>
                     <paramterType>null</paramterType>
              </paramterList>
              <returnType>integer</returnType>
           </action>
       </actionList>
</context-service>
```

Figure 6.21 - Creating a Nibble location service interface

Once an interface description exits, an application developer needs to create a `ContextService` Class that inherits the properties of the generic `BaseContextService` class and takes as a parameter the location of the XML file description, as a URL. This will then enable the properties of service to be advertised and a description made available to system entities. A running instance can be created as follows (see figure 6.22). In essence, the class constructor takes as a

parameter a URL that represents an XML description of the service available. This file is then parsed and a Java object representing the context service created. Following this, several threads of execution are created which begin advertising the services to the rest of a system and listen for incoming service requests or subscriptions.

```
// Create a Location Context Service (e.g. Nibble Location Service)
ContextService gps = new ContextService("file:///d:/gps.xml");

……….
// Class constructor
thisService = new ContextService(stringParam);

// public void run()
{
  // Parse XML description and create service state table
  thisService.parse()

  // Begin Service Aanoucements
  thisService.announce();

  // Listen for requests
  thisService.receive ();
}
```

Figure 6.22 - Instantiating a context service

Finally, a developer needs to implement the service specific communication to obtain data from the Nibble Service itself. Therefore, a method call to the `getLocation()` method of the Nibble Context Service invokes a `thisNibble.getLocationName()` call of the `getDescription()` method of the Nibble location system. Once executing, the service is available for requests and able to deliver context updates to interested (i.e. registered) client applications. This is achieved by periodically invoking the `thisNibble.getLocationName()` method (e.g. every 3 seconds).

6.3.5.3 Analysis

The level of abstraction (indirection) created by deploying a location context service ensures that any changes to the underlying Nibble location system do not adversely affect the use of the service by client applications. Furthermore, it would be possible to modify the Nibble context service to make use of any technology to obtain the location information while the service interface remains unchanged since this is transparent to the client applications. Furthermore, the use of a context service

facilitates the storage of context updates and so client applications are able to query context stored.

6.3.6 Summary

This section has introduced a number of real scenarios that allow the key features of the context-service based architecture to be evaluated using the GUIDE II prototype applications. More precisely, each of the scenarios described above demonstrate the overall suitability of the approach detailed in this thesis when applied to a complex mobile context-aware system. Furthermore, it can be seen by directly comparing the GUIDE prototypes that the use of the CSP better facilitates execution in a dynamic environment. First, the relative ease with which a complex context-aware system, such as the GUIDE system, can be readily extended and modified through the use of context services was demonstrated. Following this, support for the introduction of new context services was explored including how the use of context constraints aids the process of context discovery. Maintaining a consistent view of the environment and supporting shared access to context was then demonstrated by showing how the CSP enables a context-aware application to maintain state between application sessions and devices. Support for disconnect operation was then evaluated and in particular how the CSP supports both synchronous and asynchronous forms of communication. Finally, extensibility was evaluated by demonstrating the support provided for designers of mobile context-aware applications. The remainder of this chapter provides an evaluation of the overall context-service based approach with respect to the requirements identified in chapter four.

6.4 Evaluation With Respect To Requirements

This section evaluates the extent to which the service based architecture presented in this thesis fulfils the requirements outlined in chapter four. For the most part, this is a relatively straightforward process, with a summary of the evaluation presented in table 6.2. However, it is necessary to supplement this with a general discussion of issues arising; accordingly this table is complemented with the following discussion which, where appropriate, refers to the above scenarios and the specific components of the architecture that address each requirement.

Requirement	Satisfied
R1: Supporting User and Device Mobility	☺
R2: Support Persistence of Application and User State	☺
R3: Support Flexible Interaction Models	☺
R4: Security and Privacy of User Data	☺
R5: Extensibility	😐
R6: Modelling the Environment	😐
R7: Management of Shared and Distributed Data	😐
R8: Configuration and Interoperability	☹
R9: Context Capture	☺
R10: Context Interpretation	😐
R11: Infrastructure Transparency	☺
R12: Context Presentation, Adaptation and Persistence	☺
R13: Ability To Support Awareness	☺
R14: Ability To Support Context Sharing	☺
R15: Specification and Representation of Context	😐

☺ = yes. 😐 = partly. ☹ = no.

Table 6.2 - Fulfilment of Requirements

First, the service based architecture fulfils requirement R1 through the use of the context service provider (CSP). This component, in essence, acts as an agent on behalf of executing applications and maintains state pertaining to the operating environment. Furthermore, it can also be regarded as a portal to the context space since it provides a representation of the context services available to an application at any given time. Furthermore, the session management features provided by the session manager component afford application-level support for disconnected operation and synchronisation of state through the replaying of any events that may have occurred during disconnection form the network. Through the provision of a layer of abstraction between the higher level applications layer and the lower level service infrastructure requirement R2 has been satisfied, since, by utilising the CSP user and application state is retrievable from the context repository from any application executing on any device.

The modifications to the communications medium introduced in chapter three enable a diverse set of communications possibilities to be achieved satisfying requirement R3. Applications are able to take advantage of an asynchronous event based (publish/subscribe) model to autonomously receive context updates, in addition to the scalable broadcast approach to data dissemination described in chapter three.

Furthermore, a synchronous request/response (client/server) model of interaction between applications and a context source also exists and includes, though the transparent session management features afforded by the CSP, improved support for disconnected operation by employing a context-based caching mechanism (based on a variety of contextual factors).

Requirement R4 is satisfied since user state is stored on a trusted central server and access strictly controlled through a user specified profile. Furthermore, since users explicitly state their context interests and specify their privacy constraints (based on the notion of user and group privileges), they have ultimate control over how their personal context is utilised and shared amongst other users. Requirement R5 is partially satisfied since application developers are free to develop and instantiate context services independently from the rest of the system. The use of context services represented in XML support the incremental development of context services without adversely affecting any client applications (requirement R6). Despite an environment model not being an implicit element within the CSP architecture, the geographic information model detailed in chapter three facilitates the geographic modelling of contextual entities through its service interface. This geographic model serves as a valuable mechanism for calculating relationships between system entities and the standard interface enables access to the navigation service to be gained from a range of applications. Furthermore, the visual editor described in [Franz,98] enables a geographic information model to be easily created and deployed.

Requirement R7 was demonstrated by the example detailed in scenario two. Here, it was shown how the CSP API enables client applications to maintain state pertaining to application sessions despite user and host mobility. Furthermore, a mobile user is able to continue operation and retrieve application state even when changing between mobile end-systems. Configuration and interoperability (requirement R8) were addressed by scenario five, and how the infrastructure could be utilised to enable a context service to be rapidly deployed and utilised by a client context-aware application. Furthermore, the CSP architecture was implemented and tested using a variety of target platforms and devices to illustrate its flexibility.

Requirement R9 was demonstrated through the use of generic service interfaces exposed to client applications. This facilitates a wide range of transparent context

capture techniques to be supported. The separation of concerns that exists allows context gathering in a bespoke manner to be hidden from an application since, to a large extent, this is irrelevant to a user of a service. Furthermore, services, both physical devices and virtual entities, are modelled in the same manner thus providing a consistent means for representing, communicating with and utilising context services. In the current implementation, the CSP makes use of the logical operators "AND", "OR", "GREATER", "LESS" and "EQUALS" in order to create semantic relationships and situations and to partially satisfy requirement R10. In addition to this, application designers should be able to specify more complex composite events or situations in order to better simulate the complex interactions which occur within the real world. This is tightly coupled with requirements R6 and R15 and discussed further as an area of future work in section 7.3.2.

Infrastructure transparency (Requirement R11) is supported through the use of technologies such as Java, XML and HTTP, since they afford device and platform independence. Context presentation, adaptation and persistence are largely application specific features. However, the context management component of the CSP includes a mechanism to store context on a per application basis, thus providing application developers with a mechanism for querying persistent context state. Furthermore, the context service abstraction provides a degree of local storage and facilitates some level of persistent state for context sensing devices (e.g. a humidity sensor).

The role of context feedback to aid environmental awareness still remains a large area for exploration; however, the use of user profiles as a mechanism to specify privacy controls in a context-aware environment has been demonstrated. Furthermore, coupled with the use of device awareness, more interactive and collaborative applications are possible since increasing the level of environment awareness increases the accuracy to which system entities, including users, can be represented. The sharing of context between executing applications was demonstrated in section 6.3.3. It was argued that a clear benefit exists if applications are able to share the context sensing infrastructure. Furthermore, less communications burden is placed on context services to manage context updates to multiple applications residing on a single device. The generic specification and representation of context entities was

partially addressed through the use of XML as a way of describing context services. However, relating to requirement R10, this is an area for future work detailed in section 7.3.2.

6.5 Summary

This chapter has described an evaluation of the context-service and the CSP detailed in chapters four and five. The context-service was evaluated using, as a test vehicle, the GUIDE II prototype application. More specifically, the evaluation sought to *qualitatively* examine the flexibility and overall effectiveness of the context service provider based on several real-world scenarios. The realisation of the GUIDE II prototypes used as a case study have been a valuable contribution to this work in itself. First, they have demonstrated the practical benefits of the approaches detailed in the previous chapters when applied to a large scale context-aware system. Second, they provided a basis for the evaluation of the architecture and for the validation of the requirements.

The author believes that, in general, the features provided through the use of context-services successfully verify these requirements, in particularly, in terms of supporting context-aware applications in dynamic and unpredictable environments. Moreover, through a direct comparison of the two GUIDE prototype implementations, it has been shown how the use of context services and the CSP has promoted rapid development and deployment of services and reduced the tight coupling originally present between an application, its functionality and the hardware infrastructure. More specifically, the abstraction focuses on computational entities and sources of contextual information as services offering context (state), properties and actions which may be discovered, utilised and manipulated dynamically by client applications. Finally, system flexibility was demonstrated by engineering and testing the architecture and client applications using a range of different target platforms.

The final chapter (chapter seven) presents the conclusions of this thesis before identifying several areas of future work, some of which are currently in progress at Lancaster and build upon the ideas presented in this thesis.

Chapter 7

Conclusions

7.1 Summary of the Thesis

This thesis has investigated the use of a context service based architecture as a mechanism to support context-aware applications destined for use in mobile environments. More specifically, this thesis has identified and considered a set of requirements for building a service to support and simplify the development of mobile context-aware applications. A prototype architecture based on the identified requirements has been introduced and detailed using the concept of context-services and a context service provider (CSP). The architecture has been evaluated using the Lancaster GUIDE system as a research vehicle, a fully operational *real* tourist application affording personalised information, navigation and interactive services to visitors to the city of Lancaster from a range of portable end-systems.

The introductory chapter presented the broad areas of research pertinent to this thesis. First, the recent advances in personal computing and communications technologies were detailed following which the terms context and context-awareness were formally introduced. It was then argued that little support currently exists for context-aware applications within the mobile computing domain and that current approaches are generally application specific, static in nature, do not suit a ubiquitous service

environment and are often confined to indoor research laboratories. Therefore, by providing flexible and extensible mechanisms to support context-aware applications, rapid prototyping, real world deployment and evaluation will eventually lead to a further understanding of the higher level issues relating to the implications associated with ubiquitous service environments.

The next chapter (chapter two) presented a survey of the two main research areas which this thesis combines. First, current support for mobility within distributed systems was examined and, in particular, approaches to supporting disconnected operation, information dissemination and techniques for dynamic discovery were described as pertinent areas of research when considering ubiquitous services. Secondly, an in-depth examination of context-aware systems and architectures was presented in relation to the context-aware life cycle: context discovery, context selection and context use [Schilit,95]. In particular, it was noted that current systems are not easily evolved and are very inflexible with regards dynamic environments.

Chapter three detailed the GUIDE system [Mitchell,98], a unique city-wide wireless testbed for conducting research into mobile and context-aware computing and used to investigate the use of context-awareness as a technique for supporting tourists. In addition, this chapter described how the GUIDE system has made possible research into a number of key areas including: large scale wireless infrastructure deployment, mobile computing and context-aware concepts evaluation, human factors and user field trial analysis. Furthermore, this initiative afforded an infrastructure on which to prototype and deploy applications outside of a laboratory environment.

Following this, chapter four explored the use of context services for supporting the development of mobile context-aware applications. Initially, a comprehensive set of requirements was derived by analysing both the key aspects detailed in chapter two with a critique of the GUIDE prototype implementation. Following this, the design of a context service based architecture was described motivated by a critique of the GUIDE implementation.

Chapter five described the implementation of a Context Service Provider (CSP) and prototype context services designed to provide application programmers with a convenient and flexible mechanism for dealing with context-aware applications

targeted at mobile environments. In essence, support for disconnected operation is achieved through a combination of session management features and data dissemination techniques enabling state to be maintained despite fluctuations in the environment. Furthermore, the context service is capable of adapting to its environment through the use of context discovery based on user stipulated context constraints. This form of discovery is more general than the proximity based solutions detailed in chapter two and enables more complex applications to be realised.

An evaluation of the context service was detailed in chapter six. The primary objective of this qualitative evaluation was to ascertain the usefulness of the features provided by the prototype context service. First, the case-study of a mobile context-aware application, GUIDE II, was presented in which the usefulness and flexibility of the CSP was demonstrated in relation to several real scenarios before, finally, the CSP was evaluated with respect to the requirements identified in chapter four.

The remainder of this chapter describes the contributions of the thesis and addresses several areas of potential future work before presenting some concluding remarks.

7.2 Contributions of the Thesis

7.2.1 Major Contributions

7.2.1.1 A Novel Architecture Supporting Mobile Context-Aware Applications

An important contribution of this thesis is the design, implementation and evaluation of an architecture capable of supporting mobile context-aware applications. The motivation for the architecture originates from the observation that both the research field and the telecommunications industry has seen a recent growth of interest in applications that exploit contextual information; however, current solutions are often application specific and not well suited to mobile environments. A significant contribution of this thesis is thus the observation that generic mechanisms are required capable of supporting the development of context-aware systems given a range of execution environments.

In support of the need for an appropriate architecture, this thesis provided a comprehensive survey of the state-of-the-art in a number of vital research domains, namely, context-aware and mobile computing, and discussed the implications of each for research into the next generation *always-on* systems. An analysis of the related work in conjunction with a critique of the GUIDE prototype implementation was used to identify the main requirements and design goals for an architecture suitable for mobile context-aware environments.

More precisely, this critique revealed that significant benefits may be gained by 'pushing' the context management process into the distributed networked environment and representing functional entities as context services. As a direct result, this immediately increases the accessibility of services by users, devices or applications within a distributed environment. Furthermore, the provision of standard interfaces coupled with the separation of low level sensors from services and applications, affords incremental development. As a result, system entities can be changed dynamically as new sensors, services and devices appear during application execution without causing any adverse affects. By sharing core-services, as detailed in chapter six, context-aware applications may be smaller in size since application functionality can reside within the infrastructure and be shared instead of applications being single monolithic and self-contained entities.

This thesis proposed a solution that aims to extend and apply the advantages of current context-aware and mobile computing principles to the development of future context-aware applications by using the concept of context-services and a CSP (context service provider) as an abstraction for providing the necessary support for dynamic environments. Access to the underlying services is controlled by the CSP which, through an explicit registration process (in which users are able to stipulate their context interests and constraints), is able to dynamically adapt to the services made available according to changes in the environmental context. Furthermore, the session management features of the CSP, including the adaptive communications protocol, enable an application to maintain access to contextual information despite rapid fluctuations in the operating environment.

7.2.1.2 Deployment of a Fully Operational City-Wide Wireless Infrastructure

The deployment of the GUIDE wireless infrastructure has enabled large scale user trials and live prototype applications to be developed. This ubiquitous networked environment has enabled a number of valuable insights to be gained relating to the deployment of context-aware applications designed for outdoor city environments. Furthermore, as a result of the initial infrastructure roll-out phase of the Lancaster GUIDE initiate, a wealth of experience and valuable lessons have been learnt including the practical issues relating to deploying wireless 802.11 networks within a wide area metropolitan environment [Schmid,01], the experiences gained from supporting a fully operational tourist system [Cheverst,00b] and, techniques suitable for the user evaluation of a context-aware system [Cheverst,00a].

A significant contribution of this thesis is therefore the development and deployment of fully operational context-aware system, the Lancaster GUIDE system. The engineering and re-engineering phases of the complex GUIDE application has enabled the benefits of the context-service based approach to be realistically demonstrated within large a *real* application domain. A further contribution of this thesis can be found in the evaluation results from the motivating scenarios detailed in chapter six. In addition to providing a foundation for the evaluation of the context-based architecture, the realisation of the prototype applications used have also contributed to the evaluation of the overall approach. The experiences gained from developing and evaluating these prototypes has enabled a blueprint for future applications to be created. The evaluation demonstrates the applicability and suitability of the approach for mobile domains.

7.2.2 Other Significant Contributions

7.2.2.1 The Development of a Contextual Information Model

The development of a contextual information model capable of supporting geographic and context-sensitive information within a dynamic execution environment presents a novel and valuable contribution of the work detailed in this thesis. The information model detailed in chapter three provides a basis for navigation in both real and virtual worlds in addition to supporting the dynamic updating of contextual attributes

received through context events. More specifically, the information model is capable of receiving a range of events which may adapt the information dynamically.

7.2.2.2 The Development of an Adaptive Context-Aware Navigation Aid

An adaptive context-aware route guidance tool capable of providing personalised tours based on a wide range of extensible contextual attributes represents a further interesting contribution to the work detailed in this thesis. The tour creation component is able to utilise context such as: visitor interests, selected city attractions, travel constraints, available time, weather and budget allowance, to achieve the correct balance between the different demands on a visitor and produce a suitable personalised city tour. Moreover, since many of the above factors vary over time, the tour guide component is able to dynamically recalculate the proposed tour during the course of a single day.

7.2.2.3 Adaptive Communications Protocol

An additional contribution to the work detailed in this thesis is the development and implementation of an adaptive communications mechanism for data dissemination. This provides support for a potentially large user community within close proximity. Furthermore, the protocol extensions detailed in chapter five, which facilitate application subscriptions to context events, provides a flexible means of communications for mobile applications which may incur rapid changes in the communications QoS. This communications protocol provides a variety of interaction styles suitable for context-aware applications designed for a wide range of execution environments.

7.2.2.4 User Field Trial Evaluation Results

The research presented in this thesis has verified the effectiveness of context-awareness as a basis for multimedia mobile computing applications within the tourism industry. As a direct result of a series of user field trial evaluations of the Lancaster GUIDE prototypes valuable user feedback has been obtained to further ascertain the end user requirements for future interactive mobile applications. Furthermore, these evaluation results have been widely published within the human factors research

community [Cheverst,00a], [Cheverst,00b], [Cheverst,01b] and have formed the basis for current work-in-progress at Lancaster (as described in sections 7.3.3 and 7.3.4).

7.3 Future Work

This thesis has concentrated on the prototype implementation of the CSP architecture and the prototype applications that are able to exploit its key features. One of the first steps to continue the research presented in this thesis must therefore be the extension of the base CSP architecture to support a wider range of applications. One of the main research efforts was to establish a suitable initial testbed for research into context-aware computing in mobile environments. In essence, this environment can then be used to experiment, (i.e. develop, deploy and evaluate) novel applications with relative ease. The following sections detail possible research areas that could further enhance the properties of the CSP architecture.

7.3.1 Unified Context Discovery Architecture

Chapters four and five described the need to support discovery in order to facilitate application execution in mobile environments. Although the development of a reliable, scalable and unifying discovery architecture remained outside the scope of this thesis, it still remains a key consideration to the success of applications within mobile or ubiquitous service environments.

Since future ubiquitous service environments will be characterised by a heterogeneous mix of services and technologies, devices and applications, users will regularly need to interact with multiple, potentially specialised, service location and device interaction technologies. The current lack of a unifying approach to discovery suited to the mobile domain suggests that this is a potentially large area of exploration. Furthermore, any proposed solutions should enable users and clients to readily discover resources and services across a range of infrastructures and devices, without requiring any *a priori* knowledge of the underlying infrastructure or the components they relate to. More specifically, any solutions should offer a scalable, efficient and, vitally, usable approach that takes into consideration user demands (i.e. constraints) to succeed in emerging ubiquitous service environments [Czerwinski,99].

7.3.2 Standardised Context Specification and Modelling

This thesis has detailed how the CSP makes use of the logical operators "AND", "OR", "GREATER", "LESS" and "EQUALS" in order to create semantic relationships and situations and to enable decision making to take place. An obvious extension to this model is to allow application designers to specify more complex composite events or situations in order to better simulate the complex interactions which occur within the real world, such as, near, far, inside, opposite. In this way, the virtual representation (i.e. both semantic and geographic model) of the world may better represent the physical and computational entities that exist.

This may involve the creation and standardisation of context representation and modelling. Some initial solutions to modelling context have been proposed in this thesis and models proposed by Leonhardt [Leonhardt,98] and Jose [Jose,01a] offer insightful approaches which could be utilised. However, future work, on GUIDE and within the research community in general, will need to investigate the adoption of standard models in order to cope with the complex relationships that exist. This effort should aim to develop schemas (perhaps XML based) and ontologies in order to derive common syntax and semantics. This may involve the following challenges:

- Identifying useful context types and data formats applicable to a wide range of context-aware applications.
- Identifying useful predicates applicable to a variety of context-aware applications and situations.
- Defining common semantic relationships, for example, "near" and "inside".
- Specifying the granularity and the scope of events.
- Providing suitable access control mechanisms to ensure user security and privacy is not compromised.

7.3.3 User Controlled and Automatic Adaptation

This thesis has focussed on automatic application adaptation based on a variety of user stipulated and environmentally sensed contextual attributes. However, future work could include the incorporation of specific adaptation mechanisms and policy

adaptation techniques such as those found in [Efstratiou,01]. It is expected that the CSP and the use of context services will accommodate such techniques and practices without significant alteration. However, the exercise of extension will be useful in verifying this presumption. One area in which the use of user created adaptation policies may prove useful is within wireless overlay networks, such as those described in [Finney,99] and currently under development as part of the Mobile IPv6 Systems Research Lab (MSRL) [MSRL,01]. This new research initiative is currently extending the GUIDE wireless infrastructure and aims to supplement high-bandwidth wireless networking [Cisco,00] with next generation telecommunications networks, such as HSCSD and GPRS. The overall project aim is to develop and test (i.e. evaluate) future network and service solutions based around the Mobile IPv6 protocol [Perkins,96], [Perkins,97]. In essence, this will draw upon many of the experiences detailed in this thesis as part of the GUIDE initiative in addition to other research projects at Lancaster such LandMarc [Landmarc,01].

Within this type of environment, applications may be required to adapt to a range of complex context attributes, for example, a user of video conferencing application [Mitchell,01] with access to multiple network interfaces may choose their service and communications channel based on awareness information relating to a remote user's device constraints and the available service providers, e.g. cost based considerations. Issues such as these again point to the need for a sufficient model capable of representing a wide range of contexts within a potentially wide area context space.

7.3.4 Trust, Security and Privacy

This thesis has highlighted the importance of security and user privacy relating to user context. In this thesis a simple username and password approach has been adopted and, furthermore, the implementation has assumed a trusted and secure network [Glass,00]. In future versions, which facilitate access to remote data services, the ability to share access to personal information will need to be investigated in more depth. In more detail, this thesis has described an access control approach based on the traditional notion of relationships [Sikkel,97], which is clearly appropriate for GUIDE, for example, only allowing the sharing of personal information between members of a family group. However, it may also be interesting to investigate the

issues raised by supporting privacy in relation to place [Bullock,97]. For example, a visitor may only be prepared to let their location be revealed to others who are in the same place because the person can implicitly trust the other people in that place. One example of such a place might be a members-only club. This notion of utilizing spatial boundaries for access control is reflected in the work on SPACE by Bullock [Bullock,97]. Consider also the situation in which a visitor might be prepared to be open to interruption when sitting in a café, but not when rushing around an art gallery that is soon to close. These simple scenarios demonstrate that there is still a great deal of potential for exploring the use of context, such as location, to aid decision making within interactive mobile applications.

In addition, a further drive of the GUIDE II project is to go beyond the notion of trusted, secure networks and to investigate the issues relating to providing public access wireless networks. Here, the idea is to open up the wireless networking infrastructure to citizens of Lancaster such that they might use a range of new applications using their own personal devices (e.g. laptops, PDAs, cell-phones, etc). It is hoped that by offering wireless connectivity to citizens a wider community of mobile users can be established and leveraged for mobile systems research, for example, supporting school children through a range of interactive educational applications such as exploring the city of Lancaster at various points in the city's historic past. An initial investigation into the area of public access networks has provided the following preliminary requirements in terms of the features offered [Friday,01b]:

- Simplicity and convenience for potential users (i.e. ease of installation).
- Fine grained access control and accounting for service providers.
- Reasonable levels of security and authentication such that users can trust the system and the system is not vulnerable to exploitation.
- Support unmodified use of legacy Internet applications.
- Scalability in terms of number of users and the extensibility of the system to cover the metropolitan area.

7.3.5 Tool Support

Among the many benefits of component-oriented software are those of semi-automated construction of applications or composite components using special tools or visual environments. The original version of the GUIDE prototype benefited immensely from the use of a graphical 'city editor' tool to construct the information models used by the system [Franz,99]. It is hoped to extend this further to provide a simple mechanism for specifying and creating context service interfaces. This work will be based around a series of XML parsers and further dictates the need for a standard representation of context services (see section 7.3.2). The use of XML as a specification language enables the creation of context-services to remain programming language agnostic. A visual tool would have great benefits in terms of rapid application prototyping, deployment and evaluation and work is currently underway to develop a prototype as part of the MSRL project [Mitchell,01].

7.4 Concluding remarks

This thesis has observed that the proliferation of personal computing devices, communications and sensing technologies are rapidly progressing the fields of mobile and context-aware computing towards the vision of ubiquitous service environments [Weiser,93]. Moreover, users are operating in highly dynamic and ubiquitous environments which, in turn, afford continuous access to personalised information and computational services from any device at any-time.

In this thesis, the author has explored this field of research and presented a mobile context-aware architecture used to support the development of prototype applications as part of the Lancaster GUIDE initiative. The design, implementation and evaluation of this fully operational (i.e. live) tourist system [TIC,01] has provided a number of valuable contributions to the field of context-aware computing. More specifically, through a combination of context discovery, awareness, and the use of user stipulated context constraints, a convenient mechanism to base the development of context-aware applications has been established. Moreover, this experimental foundation has been created enabling further exploration into the fields of mobile and context-aware computing within the context of next generation wireless networks [Mitchell,01].

The key conclusion of this thesis is that only by deploying and using context-aware systems in real situations can a better understanding and a true appreciation of the associated implications at the many different levels be realised. Therefore, through a process of deployment and user evaluation we can, as a community, hope to gain a better understanding of how users will use mobile applications in the future.

References

[Abowd,97] Abowd, G.D., Dey, A.K., Orr, R. and J. Brotherton, "Context-awareness in Wearable and Ubiquitous Computing", Technical Report, Graphics, Visualization, and Usability Centre, College of Computing, Georgia Institute of Technology, USA. 1997.

[Abowd,99] Abowd, G.D., and Mynatt, E.D., "Charting Past, Present and Future Research in Ubiquitous Computing", ACM Transactions on Computer-Human Interaction, Special issue on HCI in the new Millennium, Vol. 7, No. 1, Pages 29-58. March 1999.

[Acharya,95a] Acharya, A., Alonso, S., Franklin, M., and Zdonik, S., "Broadcast Disks: Data Management for Asymmetric Communication Environments", In ACM SIGMOD, San Jose, CA USA, June 1995.

[Acharya,95b] Acharya, A., Badrinath, B. R., Imielinski, T., and Navas, J. C., "A WWW-Based Location-Dependent Information Service for Mobile Clients", Technical Report, Rutgers University, July 1995.

[Affective,00] MIT Media Lab, Affective Computing Research Projects Home Page. http://www.media.mit.edu/affect/

[Allanson,00] Allanson, J., "Supporting the Development of Electrophysiological Interactive Computer Systems," Ph.D. Thesis, Computing Department, Lancaster University, UK, June 2000.

[Alvarion,01] Breezecom Breezenet Product Homepage. http://www.alvarion.com/

[Apple,98] The Apple Newton PDA, http://www.planetnewton.com/

[AT&T,00] AT&T Laboratories Cambridge. The Active Badge System, 2000. http://www.cam-orl.co.uk/ab.html.

[Backhouse,92] Backhouse, A., and Drew, P., "The Design Implications of Social Interaction in a Workplace Setting", Planning and Design, Pages 573-584. 1992.

[Bahl,00] Bahl, P. and Padmanabhan, V. N., "RADAR: An In-building RF-Based User Location and Tracking System", In IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, volume 2, pages 775-84, March 2000.

[Baker,96] Baker, M. G., Zhao, X., Cheshire, S. and Stone, J., "Supporting Mobility in Mosquito Net", In 1996 USENIX Symposium on Mobile and Location Independent Computing, San Diego, CA, January 1996.

[Bederson,95] Bederson, B,. "Audio Augmented Reality: A Prototype Automated Tour Guide", In the Proceedings of the 1995 ACM Conference on Human Factors in Computing Systems (CHI '95), pp. 210-211, Denver, CO, ACM. May 7-11, 1995.

[Bennet,94] Bennett, F., Richardson, T., Harter, A., "Teleporting - Making Applications Mobile", Proceedings of 1994 Workshop on Mobile Computing Systems and Applications (WMCSA), Santa Cruz, December 1994.

[Bennet,97] Bennet, F., Clarke, D., Evans, J. B., Hopper, A., Jones, A. and Leask, D., "Piconet: Embedded Mobile Networking", IEEE Personal Communications, 4(5) pages 42-47, 1997.

[Benyon,97] Benyon, D. R. and Höök, K., "Navigation in Information Space", Hammond, J. (ed.) Proceedings of Interact 97, London: Chapman and Hall (1997).

[Berners-Lee,94] Berners-Lee,T., Masinter, L., McCahill, M., "Request for Comments: 1738 (RFC 1738), Uniform Resource Locators (URL)", available at http://www.w3.org/Addressing/rfc1738.txt

[Bettstetter,00] Bettstetter, C. and C. Renner. "A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol", In Proceedings of Sixth EUNICE Open European Summer School, Netherlands. September 2000.

[Blair,97] Blair, G. S. and Stefani, J.B., "Open Distributed Processing and Multimedia", Addison-Wesley, 1997.

[Bluetooth,99a] Bluetooth Consortium, "Specification of the Bluetooth System: Volume 1", Technical Report Version 1.0 B, Dec 1st 1999.

[Bluetooth,99b] Bluetooth Consortium, "Specification of the Bluetooth System: Volume 2", Profiles of the Bluetooth System. Technical Report Version 1.0 B, Dec 1st 1999.

[Box,00] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Frystyk, H., Thatte, S. and Winer, D, "Simple Object Access Protocol (SOAP) 1.1", Protocol specification available from http://www.w3.org/TR/SOAP/

[Braley,00] Braley, R. C., Gifiord, I. C. and Heile, R. F., "Wireless Personal Area Networks: An Overview of the IEEE P802.15 Working Group", Mobile Computing and Communications Review, 4(1), pages 26-33, 2000.

[Brown,96] Brown, P.J., "The Stick-e Document: A Framework For Creating Context-aware Applications", In the Proceedings of the Electronic Publishing, pp. 259-272, Laxenburg, Austria, IFIP. September 1996.

[Brown,97] Brown, P.J., Bovey, J. D. and Chen, X., "Context-Aware Applications: From the Laboratory to the Marketplace", IEEE Personal Communications, 4(5): pages 58-64, October 1997.

[Brown,98a] Brown, P.J., "Some Lessons for Location-Aware Applications". In First workshop on Human Computer Interactions for mobile devices (Mobile HCI), pages 58-63, Glasgow University, May 1998.

[Brown,98b] Brown, P.J., "Triggering information by context", Springer-Verlag, Personal Technologies 2(1), : pp. 1-9. September 1998.

[Brown,00a] Brown, P.J. and Jones, G.J.F., "Context-Aware Retrieval: Exploring a New Environment for Information Retrieval and Information Filtering", Personal Technologies, Springer, In Press. 2000.

[Brown,00b] Brown, P.J., Burleston, W., Lamming, M., Rahlff, O., Romano, G., Scholz, J. and Snowdon, D., "Context-awareness: Some Compelling Applications", International Symposium on Handheld and Ubiquitous Computing (HUC'00), Submitted for Publication. 2000.

[Brumitt,00a] Brumitt, B., Meyers, B., Krumm, J., Kern, A., and Shafer, S., "EasyLiving: Technologies for Intelligent Environments", In Hans-W Gellersen and P. Thomas, editors, HUC2000, Second International Symposium on Handheld and Ubiquitous Computing, Bristol, UK, September 25-27, 2000. Published by Springer-Verlag as Lecture Notes in Computer Science, vol. 1927, pages 12-29.

[Brumitt,00b] Brumitt, B. and Shafer, S. "Better Living Through Geometry", CHI Workshop on Situated Interaction in Ubiquitous Computing, April 2000. Also submitted to Springer Journal of Personal Technologies.

[BT,01] British Telecommunications plc. BT Open World Product Web Site. http://www.btopenworld.com/

[Bullock,97] Bullock, A. and Benford S., "Access Control in Virtual Environments", Symposium on Virtual Reality Software and Technology 1997 (VRST'97) Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland. (1997).

[Byun,01] Byun., H., Cheverst, K., Mitchell, K., "Achieving Context Interoperability: A Multi-layered DTD Approach", Internal Technical Report.

[Casio,01] Casio GPS Wristwatch. http://www.casio.com/watches/

[Casio,99] Casio Cassiopeia Product Home page. http://www.casio.com/personalpcs/

[Castro,01] Castro, P., Chiu, P., Kremenek, T. and Muntz, R., "A Probabilistic Room Location Service for Wireless Networked Environments", in Proc. UbiComp 2001, September 30 - October 2, 2001, Sheraton Colony Square Hotel, Atlanta, Georgia.

[Caswell,00] Caswell, D. and Debaty, P., "Creating Web Representations for Places", In Hans-W Gellersen and P. Thomas, editors, HUC2000, Second International Symposium on Handheld and Ubiquitous Computing, Bristol, UK, September 25-27, 2000. Published by Springer-Verlag as Lecture Notes in Computer Science, vol. 1927, pages 114-126.

[Chen,99] Chen, D., Schmidt, A., Gellersen, H.W., "An Architecture for Multi-Sensor Fusion in Mobile Environments", In Proceedings International Conference on Information Fusion, Sunnyvale, CA, USA, July 1999.

[Cheverst,98] Cheverst, K., Davies, N., Mitchell, K., Friday, A., "Design of an Object Model for a Context-Sensitive Tourist Guide", Proceedings of the IMC'98 Workshop on Interactive Applications of Mobile Computing, Rostock, Germany, November 1998.

[Cheverst,99a] Cheverst, K., Davies, N., Mitchell, K., Friday, A., "The Role of Connectivity in Supporting Context-Sensitive Applications". In Hans-W Gellersen, editor, HUC99, International Symposium on Handheld and Ubiquitous Computing, Karlsruhe, Germany, September 27-29 1999. Published by Springer-Verlag as Lecture Notes in Computer Science, vol. 1707, pages 193-207.

[Cheverst,99b] Cheverst, K., "Development of a Group Service to Support Collaborative Mobile Groupware", Ph.D. Thesis, Computing Department, Lancaster University. 1999.

[Cheverst,00a] Cheverst K., Davies N., Mitchell K., Friday A. and Efstratiou C., "Developing Context-Aware Electronic Tourist Guide: Some Issues and Experiences", Proceedings of CHI'2000, Netherlands, (April 2000), pp. 17-24.

[Cheverst,00b] Cheverst K., Davies N., Mitchell K. & Friday A., "Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project", Proceedings of MOBICOM'2000, Boston, ACM Press, August 2000, pp. 20-31.

[Cheverst,00c] Cheverst K., Davies N., Mitchell K. & Smith P., "Providing Tailored (Context-Aware) Information to City Visitors", Proceedings of International

Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, AH'2000, Trento, Springer-Verlag LNCS, August 2000, pp. 7-85.

[Cheverst,00d] Cheverst, K., Mitchell, K., Davies, N., Friday, A., "Sharing (Location) Context to Facilitate Collaboration Between City Visitors", Proceedings of Workshop on Interactive Applications of Mobile Computing (IMC), Rostock, Germany, November 2000.

[Cheverst,01a] Cheverst, K,. Mitchell, K., Smith, G., Davies, N., "Exploiting Context to Support Social Awareness and Social Navigation", in Proceedings of the Workshop on 'Awareness and the WWW' at CSCW '00, Philadelphia, 2nd Dec 2000.

[Cheverst,01b] Cheverst, K., Smith, G., Davies, N., Mitchell, K. and Friday, A., "The role of shared context in supporting cooperation between city visitors", Computers and Graphics, Vol. 25, No. 4 (2001) 555-562.

[Cheverst,01c] Cheverst, K., Davies, N., Mitchell, K. and Efstratiou, C., "Using Context as a Crystal Ball: Rewards and Pitfalls", Personal Technologies Journal, Vol. 3 No5 (2001) 8-11.

[Cheverst,01d] Cheverst K., Davies N., Mitchell K. and Smith P., "Exploring Context-Aware Information Push", in Proceedings of Third International Workshop on Human Computer Interaction with Mobile Devices (Mobile HCI), 10 Sept 2001, At IHM-HCI 2001, Lille, France.

[Cheverst,01e] Cheverst K., Davies N., Mitchell K. Davies., "The Role of Adaptive Hypermedia in a Context-Aware Tourist GUIDE", Communications of the ACM. To appear.

[Cisco,01] Cisco Systems Inc. Aironet. http://www.aironet.com/

[Compaq,01] The Compaq iPAQ Handheld Devices Product Home Page. http://www.compaq.com/products/handhelds

[Coyle,97] Coyle, M., Shekhar, S., Liu, D., and Sarkar, S., "Experiences with Object Data Models in Geographic Information Systems", Internal Technical Report, Department of Computer Science, University of Minnesota, U.S. 1997.

[Czerwinski,99] Czerwinski, S. E., Zhao, B. Y., Hodes, T. D., Joseph, A. D. and Katz., R. H., "An Architecture for a Secure Service Discovery Service", In Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking - Mobicom 99, pages 24-25, Seattle, Washington, USA, August, 15-20 1999.

[Dana,98] Dana, P.H., "Global Positioning System Overview", The Geographer's Craft Project, Department of Geography, The University of Texas, USA. 1998.

[Davies,94] Davies, N., Blair, G., Cheverst, K., and Friday, A., "Supporting Adaptive Services in a Heterogeneous Mobile Environment". In Luis-Felipe Cabrera and Mahadev Satyanarayanan, editors, Workshop on Mobile Computing Systems and Applications. (WMCSA), pages 153-157, Santa Cruz, CA, U.S., December, 1994. IEEE Computer Society Press.

[Davies,98a] Davies, N., Mitchell, K., Cheverst, K. and Blair, G.S., "Developing a Context Sensitive Tourist Guide", Proc First Workshop on Human Computer Interaction for Mobile Devices, Glasgow. March 1998.

[Davies,98b] Davies, N., Friday, A., Wade, S., Blair, G., "L2imbo: A Distributed Systems Platform for Mobile Computing", ACM Mobile Networks and Applications (MONET), Special Issue on Protocols and Software Paradigms of Mobile Networks, Volume 3, Number 2, August 1998, pp143-156.

[Davies,99] Davies, N., Cheverst, K., Mitchell, K. and Friday, A., "Caches in the Air: Disseminating Tourist Information in the Guide System", In Second IEEE Workshop on Mobile Computer Systems and Applications, New Orleans, Louisiana, 25-26 February 1999.

[Davies,01] Davies, N., Cheverst, K., Mitchell, K. and Efrat, A., "Using and Determining Location in a Context-Sensitive Tour Guide", IEEE Computer Journal, August 2001 (Vol. 34, No. 8) pages 35-41.

[Demers,94] Demers, A., Petersen, K., Spreitzer, M., Terry, D., Theimer, M.M. and Welch, **,** "The Bayou Architecture: Support for Data Sharing Among Mobile Users", Proc Workshop on Mobile Computing Systems and Applications, IEEE. December 1994.(MCSA), Santa Cruz, California, U.S., December 1994. Editor: Luis-Felipe Cabrera and Mahadev Satyanarayanan, IEEE Computer Society Press, Pages 2-7.

[Dey,97] Dey, A.K., and Abowd, G.D., "CyberDesk: The Use of Perception in Context-aware Computing", In the Proceedings of the 1997 Workshop on Perceptual User Interfaces (PUI '97), pp. 26-27, Banff, Alberta. October 19-21, 1997.

[Dey,98] Dey, A.K., Abowd, G.D. and Wood, A., "CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services", Knowledge Based Systems 11(1): pp. 3-13. September 30, 1998.

[Dey,99a] Dey, A.K., Futakawa, M, Salber D. and Abowd, G.D. "The Conference Assistant: Combining Context-Awareness with Wearable Computing", In the Proceedings of the 3rd International Symposium on Wearable Computers (ISWC'99), pp. 21-28, San Francisco, CA, IEEE. October 20-21, 1999.

[Dey,99b] Dey, A.K., Salber, D., Abowd, G.D., "A Context-based Infrastructure for Smart Environments", In the Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments (MANSE '99), pp. 114-128, Dublin, Ireland, Springer Verlag. December 13-14, 1999.

[Dey,00a] Dey, A.K., and Abowd, G.D., "CybreMinder: A Context-Aware System for Supporting Reminders", In the Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing (HUC2K), pp. 172-186, Bristol, UK, Springer-Verlag. September 25-27, 2000.

[Dey,00b] Dey, A.K., and Abowd, G.D., "Towards A Better Understanding of Context and Context-Awareness. In the Workshop on the What, Who, Where, When and How of Context-Awareness", affiliated with the 2000 ACM Conference on Human Factors in Computer Systems (CHI 2000), The Hague, Netherlands. April 1-6, 2000.

[Dey,00c] Dey, A.K., "Providing Architectural Support for Building Context-Aware Applications". Ph.D. Thesis, Georgia Institute of Technology, October 2000.

[Dey,01] Dey, A.K., Salber, D. and Gregory D. Abowd, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", Human-Computer Interaction 16.

[Digianswer,00] Digianswer Bluetooth Homepage. http://www.digianswer.com/

[Dix,95] Dix, A., "Cooperation Without (Reliable) Communication", In Proceedings IEE Symposium on Mobile Computing and its Applications, England, Vol. 95, No.219, Pages 41-44. November 1995.

[Dix,99a] Dix, A., Ramduny, D., Rodden, T. and Davies, N., "Places to Stay on the Move: Software Architectures for Mobile User Interfaces", Personal Technologies. 1999.

[Dix,99b] Dix, A., Rodden, T., Davies, N., Trevor, J., Friday A. and Palfreyman, P., "Exploiting Space and Location as a Design Framework for Interactive Mobile Systems", ACM Transactions on Computer-Human Interaction (TOCHI). 1999.

[Dourish,92] Dourish, P. and Bellotti, V., "Awareness and Coordination in Shared Workspaces", In Proceedings of ACM Conference on Computer-Supported Cooperative Work (CSCW '92), Toronto, Canada, pages 107-114. 1992.

[Edwards,99] Edwards, W. K. "Core Jini", Prentice Hall., Addison-Wesley. 1999.

[Efstratiou,00] Efstratiou, C., Cheverst, K, Davies, N. and Friday, A., "Architectural Requirements for the Effective Support of Adaptive Mobile Applications", IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing, 4th - 8th April 2000, IBM Palisades Executive Conference Center, Hudson River Valley (near New York City), New York, USA

[Eldridge,00] Eldridge, M., Lamming, M., Flynn, F., Jones C., and Pendlebury, D., "Satchel: Providing Access to Any Document, Any Time, Anywhere", in:

Transactions on Computer-Human Interaction, Special Issue entitled "Beyond the Workstation: Human Interaction with Mobile Systems", 2000.

[EMC,01] Market Intelligence for the World's Wireless Industry Homepage. http://www.emc-database.com/website.nsf/index/pr000714

[Ericsson,00] Ericsson Mobile Positioning System. http://www.ericsson.com/mps

[Ericsson,01] Ericsson Bluetooth Homepage. http://www.ericsson.com/Bluetooth

[Factoid,01] The Compaq Western Research Lab Factoid Project homepage. Hoe http://www.research.compaq.com/wrl/projects/Factoid/index.html

[Fels,98] Fels S., Sumi, Y, Etani, T., Simonet, N., Kobayashi, K. and Mase., M, "Progress of C-MAP: A Context-Aware Mobile Assistant", Proc. AAAI Symposium on Intelligent Environments, Technical Report SS-98-02 (1998)

[Finney,96a] Finney, J. and Davies, N., "FLUMP - The FLexible Ubiquitous Monitor Project", Proceedings of the 3rd Cabernet Radicals Workshop, Connemara, May 1996.

[Finney,96b] Finney, J. and Davies, N., "The FLexible Ubiquitous Monitor Project", Proceedings of the Third Computer Networks Symposium, July 1996.

[Finney,99] Finney, J., "Supporting Continuous Multimedia Services in Next Generation Mobile Systems", Ph.D. Thesis, Computing Department, Lancaster University. 1999.

[Flynn,00] Flynn, F., Pendlebury, D., Jones, C., Eldridge, M. and Lamming, M., "The Satchel System Architecture: Mobile Access to Documents and Services", Mobile Networks and Applications (MONET) 2000, No. 4, Vol. 5, 2000, pp 243-58.

[Forman,94] Forman, G.H. and Zahorjan, J., "The Challenges of Mobile Computing", IEEE Computer, Vol. 27, No. 6. April 1994.

[Fox,98] Fox, A., Goldberg, I., Gribble, S. D., Polito, A. and Lee, D. C., "Experience with Top Gun Wingman: A proxy-based graphical web browser for the Palm Pilot

PDA", In IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware '98), Lake District, UK, September 15--18 1998.

[Franklin,98] Franklin, M. and Zdonik, S., " 'Data in Your Face': Push Technology in Perspective", ACM SIGMOD Intl. Conference on Management of Data (SIGMOD 98) Seattle, WA, June, 1998.

[Franz, 98] Franz, M., "Creating Context-Sensitive City Models: A City Editor", Diploma Thesis at the Institute of Telematics, University of Karlesruhe, Germany.

[Friday,96] Friday, A., "Infrastructure Support for Adaptive Mobile Applications", Ph.D. Thesis, Computing Department, Lancaster University, Bailrigg, Lancaster, LA1 4YR, U.K., September 1996.

[Friday,97] Friday, A., Davies, N., Wade, S. and Blair, G., "Limbo: A Tuple Space Based Platform for Adaptive Mobile Applications", Proceedings of the International Conference on Open Distributed Processing/Distributed Platforms (ICODP/ICDP '97), Toronto, Canada, 27-30 May 1997, pp291-302.

[Friday,98] Friday, A., Davies, N., Wade, S. and Blair, G., "L2imbo: A Distributed Systems Platform for Mobile Computing", ACM Mobile Networks and Applications (MONET), Special Issue on Protocols and Software Paradigms of Mobile Networks, Volume 3, Number 2, August 1998, pp143-156.

[Friday,99] Friday, A., Davies, N., Blair, G. and Cheverst, K., "Developing Adaptive Applications: The MOST Experience", Journal of Integrated Computer-Aided Engineering, Volume 6, Number 2, 1999, pp143-157.

[Friday,01a] Friday, A., Davies, N. and Catterall, E., "Supporting Service Discovery, Querying and Interaction in Ubiquitous Computing Environments", 2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access, Santa Barbara, CA, May 20, 2001. pp. 7-13.

[Friday,01b] Friday, A., Wu, M., Schmid, S., Finney, J., Cheverst, K., Davies, N, "A Wireless Public Access Infrastructure for Supporting Mobile Context-Aware IPv6 Applications", The First Workshop on Wireless Mobile Internet (Satellite event of 7th

Annual International Conference on Mobile Computing and Networking), Rome, Italy, July 21, 2001. pp. 11-18

[Fujitsu,98] Fujitsu ICL, Teampad 7600. http://www.fjicl.com/TeamPad/teampad76.htm.

[Fujitsu,01] Fujitsu ICL Stylistic 1200 Product Specifications Homepage. http://www.fujitsupc.com/www/support.shtml?support/techspecs/pentabs/Stylistic120 01999.

[Garmin,01] Garmin International GPS Receivers. http://www.garmin.com/

[Glass,00] Glass, S., Hiller, T., Jacobs, S. and Perkins, C., "Mobile IP Authentication, Authorization, and Accounting Requirements", RFC 2977, October 2000.

[Goland,99] Goland, Y. Y., Cai, T, Leach, P., Gu, Y. and Albright, S., "Simple Service Discovery Protocol 1.0: Operating without an Arbiter", Internet-draft draft-cai-ssdp-v1-03.txt, work in progress, October 28 1999.

[Goland,00] Goland, Y. Y. and Schlimmer, J. C., "Multicast and Unicast UDP HTTP Messages", Internet-draft draft-goland-http-udp-04.txt, work in progress, October, 2 2000.

[GPRS,98] Rysavy, P., "General Packet Radio Service (GPRS)", GSM Data Today online journal, http://www.gsmdata.com/paprysavy.html. September 1998.

[GPS,01] All about GPS. Trimble home page available at http://www.trimble.com/gps/

[Gray,96] Gray, R. S., "Agent Tcl: A Flexible and Secure Mobile-Agent System", Proceedings of the 1996 Tcl/Tk Workshop, pages 9-23, July 1996.

[Gray,97] Gray, P. and Salber, D., "Modelling Space for Location-Dependent Tasks: Why Location-Independent Computing Isn't", In Ubiquitous Computing: The Impact on Future Interaction Paradigms and HCI Research, XEROX, March 23-24 1997.

[GSM,00] GSM Association. “An Introduction to the Short Message Service”, 2000. http://www.gsmworld.com/technology/sms_success.html.

[GUIDE,99] The Lancaster GUIDE Project Home Page. http://www.guide.lancs.ac.uk/

[Guttman,99a] Guttman, E., Perkins, C. and Kempf, J., “Service Templates and Service Schemes”, RFC 2609, June 1999.

[Guttman,99b] Guttman, E., Perkins, C., Veizades, J. and Day, M., “Service Location Protocol”, Version 2. RFC 2608, June 1999.

[Haartsen,98] Haartsen, J., Naghshineh, M., Inouye, J., Joeressen, O. J. and Allen, W., “Bluetooth: Vision, Goals and Architecture”, Mobile Computing and Communications Review, 2(4):28-37, 1998.

[Harter,99] Harter, A., Hopper, A., Steggles, P., Ward, A. and Webster, P., “The Anatomy of a Context-Aware Application”, In Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking - MobiCom 99, pages 59-68, Seattle, Washington, USA, August, 15-20 1999.

[Healey,00] Healey, J. and Picard, R., “SmartCar : Detecting Driver Stress”, Proc. ICPR'00, Barcelona, Spain, May 2000.

[Hodes,97] Hodes T., Katz, R., Servan-Schreiber, E. and Rowe, L., “Composable Ad Hoc Mobile Services for Universal Interaction”, In 3rd ACM/IEEE International Conference on Mobile Computing and Networking - MOBICOM97, Budapest, Hungary, 26-30 September 1997. ACM.

[Hong,01] Hong, J. I. and LandayAn, J. A., "Infrastructure Approach to Context-Aware Computing", Technical Report.

[HP,01] HP Labs. CoolTown Project Web Site, 2001. http://www.cooltown.hpl.hp.com.

[Hull,97] Hull, R., Neaves, P. and Bedford-Roberts, J., "Towards Situated Computing", In the Proceedings of the 1st International Symposium on Wearable Computers (ISWC'97), pp. 146-153, Cambridge, MA, IEEE. October 13-14, 1997.

[Ibutton,00a] Dallas Semiconductors iButton Product Homepage. http://www.ibutton.com/ibuttons/index.html

[Ibutton,00b] Dallas Semiconductors Weather Instruments Product Homepage. http://www.ibutton.com/weather/index.html

[Imielinski,94] Imielinski, T. and Viswanathan, S., "Adaptive Wireless Information Systems", In SIG in Database Systems Conference, Tokyo, Japan, 1994.

[IrDA,98] Infrared Data Association. IrDA - The Infrared Data Association Web Site, 1998. http://www.irda.org.

[Izadi,00a] Izadi, S., "Infrastructure Support for Ubiquitous Collaboration", Ph.D. Internal Report, September 2000.

[Izadi,00b] Izadi, S., Coutinho, P., Rodden, T., Smith, G. "The FUSE platform: Supporting Ubiquitous Collaboration within Diverse Mobile Environments". Journal of Automated Software Engineering. Kluwer Press. To Appear.

[Izadi,01b] Izadi, S., Fraser, M., Benford, S., Flintham, M., Greenhalgh, C., Rodden, T., and Schnädelbach, H., "Citywide: supporting interactive digital experiences across physical space", In Proc. Mobile HCI'01.

[Jacobsen,97] Jacobsen, K., Johansen., D., "Mobile Software on Mobile Hardware - Experiences with TACOMA on PDAs", Technical Report 97-32, Department of Computer Science, University of Tromsø, Norway, December 1997.

[Jacobsen,99] Jacobsen, K., Johansen, D., "Ubiquitous Devices United: Enabling Distributed Computing Through Mobile Code", ACM Symposium on Applied Computing 1999 (SAC '99.), pages 399-404.

[Johansen,95] Johansen, D., Renesse, R. and Schneider, F. B., "An Introduction to the TACOMA Distributed System", Department of Computer Science, University of Tromsø, Norway, June 1995, Technical Report 95-23

[Jose,99] Jose, R. and Davies, N., "Scalable and Flexible Location-Based Services for Ubiquitous Information Access", In Hans-W Gellersen, editor, HUC99, International Symposium on Handheld and Ubiquitous Computing, Karlsruhe, Germany, September 27-29 1999. Published by Springer-Verlag as Lecture Notes in Computer Science, vol. 1707, pages 52-66.

[Jose,01a] Jose, R., An Open Architecture for Location-Based Services in Heterogeneous Mobile Environments, PhD Thesis, Computing Department, Lancaster University. 2001.

[Jose,01b] Jose, R., Moreira, A., Meneses, F. and Coulson, G., "An Open Architecture for Developing Mobile Location-Based Applications over the Internet", In 6th IEEE Symposium on Computers and Communications, Hammamet, Tunisia, 3-5 July 2001.

[Joseph,95] Joseph, A., DeLespinasse, A., Tauber, J., Gifford, D. and Kaashoek, M.F., "Rover: A Toolkit for Mobile Information Access", Proc. 15th ACM Symposium on Operating System Principles (SOSP), Copper Mountain Resort, Colorado, U.S., ACM Press, Vol. 29, Pages 156-171. 3-6 December 1995.

[Katz,94] Katz, R. H., "Adaptation and Mobility in Wireless Information Systems", IEEE Personal Communications, 1(1):6-17, 1994.

[Katz,96] Katz, R. H., Brewer, E., Amir, E., Balakrishnan, H., Fox, A., Gribble, S., Hodes, T., Jiang, D., Nguyen, G., Padmanabhan, V. and Stemm, M, "The Bay Area Research Wireless Access Network (BARWAN)", In 41st IEEE Computer Society International Conference.(IEEE COMPCON), pages 1-12, Santa Clara, California, U.S., February 1996. IEEE Press.

[Kenelec,01] Kenelec Scientific Products Homepage. http://www.kenelec.com/

[Kidd,99] Kidd, C. D., Orr, R. J., Abowd, G. D., Atkeson, C. G., Essa, I. A., MacIntyre, B., Mynatt, E., Starner, T. E. and Newstetter, W., "The Aware Home: A Living Laboratory for Ubiquitous Computing Research", Proc. of the Second International Workshop on Cooperative Buildings (CoBuild'99), October 1999.

[Kindberg,00] Kindberg, T., Barton, J., Morgan, J., Becker, G., Bedner, I., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Pering, C., Schettino, J., Serra, B., Spasojevic, M, "People, Places, Things: Web Presence for the Real World", Presented at WMCSA December 7-8, 2000, Monterey, California, USA.

[Kistler,91] Kistler, J. J. and Satyanarayana, M, "Disconnected Operation in the Coda File System", Proc. 13th ACM Symposium on Operating Systems Principles (SOSP), Asilomar Conference Center, Pacific Grove, U.S., ACM Press, Vol. 25, Pages 213-225. 13-16 October 1991.

[LandMarC,00] The LandMarC Project Home Page, 2000, http://www.landmarc.net/

[Lamming,94] Lamming, M. and Flynn, M., "Forget-me-not: Intimate computing in support of human memory", In the Proceedings of the FRIEND 21: International Symposium on Next Generation Human Interfaces, pp. 125-128, Meguro Gajoen, Japan. 1994.

[Leonhardt,98] Leonhardt, U., Supporting Location-Awareness in Open Distributed Systems. Ph.D. Thesis, Imperial College of Science, Technology and Medicine, University of London, 1998.

[Long,96] Long, S., Kooper, R., "Rapid Prototyping of Mobile Context-Aware Applications: The Cyberguide Case Study", Proc. 2nd ACM International Conference on Mobile Computing (MobiCom), Rye, ACM Press, New York, 1996, pp 97-107.

[Marmasse,00] Marmasse N. and Schmandt, C., "Location-Aware Information Delivery with ComMotion", Proc. Symposium on Handheld and Ubiquitous Computing, Bristol, UK. (2000)

[Meyer,98] Meyer, D., "Administratively Scoped IP Multicast", RFC 2365, July 1998. http://www.faqs.org/rfc/rfc2365.txt.

[Meyers,00] Meyers, B. and Kern, A. "<Context-Aware> schema </Context-Aware>", CHI Workshop on The What, Who, When, Where, Why, and How of Context-Awareness, Affiliated with 2000 ACM Conference on Human Factors in Computer Systems (CHI 2000), The Hague, Netherlands. April 1-6, 2000.

[Microsoft,99] Microsoft Corporation. Universal Plug and Play Forum Resources Homepage. UPnP Device Architecture Specification. Technical Report Version 0.90, 10 November 1999. http://www.upnp.org/resources.htm

[Microsoft,01a] Microsoft Corporation. Windows CE Web Site, 2001. http://www.microsoft.com/windows/embedded/ce/.

[Microsoft,01b] Microsoft Corporation. Pocket PC Web Site, 2001. http://www.microsoft.com/mobile/pocketpc/.

[MIT,95] Massachusetts Institute of Technology. MIT Smart Rooms, 1995. http://www.white.media.mit.edu/vismod/demos/smartroom/.

[Mitchell,98] Mitchell, K, "Developing a Context Sensitive Tourist Guide", Proceedings of the Fifth Cabernet Radicals Workshop, Valadares, Porto, Portugal, July 1998.

[Mitchell,01] Mitchell, K., Cheverst, K., Davies, N., "Applications for City Visitors and Residents : A Proposal for the Mobile IPv6 Systems Research Lab", Internal Technical Report, August 2001, available online at http://www.mobileipv6.net/

[MSRL,01] The Mobile IPv6 Systems Research Laboratory. Project Homepage available at http://www.mobileipv6.net/

[Mummert,95] Mummert, L., Ebling, M. and Satyanarayanan., M., "Exploiting Weak Connectivity for Mobile File Access,", Proc. 15th ACM Symposium on Operating System Principles (SOSP), Copper Mountain Resort, Colorado, U.S., ACM Press, Vol. 29, Pages 143-155. 3-6 December 1995.

[Mynatt,98] Mynatt, E. D., Back, M,. Want, R., Baer, M. and Ellis, J. B., "Designing Audio Aura", In the Proceedings of the CHI '98 Conference on Human Factors in Computing Systems, pp. 566- 573, Los Angeles, CA, ACM. April 18-23, 1998.

[Nelson,98] Nelson, G., "Context-Aware and Location Systems", Ph.D. Thesis, University of Cambridge, Computer Laboratory, England. May 1998.

[Noble,97] Noble, B. D., Satyanarayanan, M., Narayanan, D., Tilton, J. E., Flinn, J. and Walker, K. R., "Agile Application-Aware Adaptation for Mobility", In Proceedings of the 16th ACM Symposium on Operating System Principles, October 1997.

[Noble,99] Noble, B.D. and Satyanarayanan, M., "Experience with Adaptive Mobile Applications in Odyssey", Mobile Networks and Applications (MONET) Vol. 4, 1999.

[Nokia,99] Nokia Bluetooth Homepage. http://www.nokia.combluetooth

[Nokia,00] Nokia Wireless LAN Homepage. http://www.nokia.com/networks/wireless_lan/

[Nokia,01] Nokia 9210 Communicator. http://www.nokia.com/phones/9210/index.html

[Oberlander,97] Oberlander J., Mellish C., O'Donnell, M., "Exploring A Gallery with Intelligent Labels", In Proceedings of the Fourth International Conference on Hypermedia and Interactivity in Museums (ICHIM97), Paris (1997)

[onCue,00] aQtive onCue. http://www.aqtive.com/consumer/oncue/oncue.html

[Oppermann,98] Oppermann, R., Specht, M., "Adaptive Support for a Mobile Museum Guide", Workshop on Interactive Application of Mobile Computing (IMC'98), Rostock, November 1998.

[Oppermann,99a] Oppermann, R., Specht, M., "A Nomadic Information System for Adaptive Exhibition Guidance", David Bearman, Jennifer Trant (eds.): Proceedings of

the International Conference on Hypermedia and Interactivity in Museums (ICHIM 99), pp. 103 - 109, Washington, September 23 - 25, 1999.

[Oppermann,99b] Oppermann, R., Specht, M., Jaceniak, I, "Hippie: A Nomadic Information System", Hans-W. Gellersen (ed.): Proceedings of the First International Symposium Handheld and Ubiquitous Computing (HUC'99), pp. 330 - 333, Karlsruhe, September 27 - 29, 1999.

[Orinoco,01] Agere's ORiNOCO (formerly Lucent WaveLAN) Product Homepage. http://www.wavelan.com/

[Orr,00] Orr, R. J., Abowd, G. D., "The Smart Floor: A Mechanism for Natural User Identification and Tracking", Proceedings of CHI'2000, Netherlands, (April 2000), The Hague, Netherlands, April 1-6, 2000.

[Palm,00] Palm Inc. Palm.com, 2000. http://www.palm.com/products/index.html.

[Pascoe,98a] Pascoe, J., "Adding Generic Contextual Capabilities to Wearable Computers", In the Proceedings of the 2nd IEEE International Symposium on Wearable Computers (ISWC'98), pp. 92-99, Pittsburgh, PA, IEEE. October 19-20, 1998.

[Pascoe,98b] Pascoe, J, Ryan, N. S. and Morse, D. R., "Human-Computer-Giraffe Interaction - HCI in the Field", In the Workshop on Human Computer Interaction with Mobile Devices, Glasgow, Scotland. May 21-23, 1998.

[Pederson,97] Pederson, E. R. and Sokoler, T., "AROMA: Abstract Representation of Presence Supporting Mutual Awareness", In the Proceedings of the 1997 ACM Conference on Human Factors in Computing Systems (CHI '97), pp. 51-58, Atlanta, GA, ACM. March 22-27, 1997.

[Perkins,96] Perkins, C., "IP Mobility Support", RFC 2002, October 1996.

[Perkins,97] Perkins, C., "Mobile IP", IEEE Communications Magazine, 1997.

[Perkins,99] Perkins, C. and Guttman, E., "DHCP Options for Service Location Protocol", RFC 2610, June 1999.

[Ponintcast,96] The Pointcast Technology Homepage. http://www.infogate.com/

[Pointsix,01] Point Six Inc. Homepage. http://www.pointsix.com/

[Randell,00] Randell, C. and Muller, H., "The Shopping Jacket: Wearable Computing for the Consumer", In Peter Thomas, editor, Personal Technologies vol.4 no.4, pages 241--244. Springer, September 2000.

[Rhodes,96] Rhodes, B. and Starner, T., "The Remembrance Agent: A Continuously Running Automated Information Retrieval System", The Proceedings of The First International Conference on The Practical Application of Intelligent Agents and Multi Agent Technology (PAAM '96), London, UK, April 1996, pp. 487-495.

[Román,00] Román, M. and Campbell, R. H., "Gaia OS: Active Spaces", In Proceedings of the 9th ACM SIGOPS European Workshop, Kolding, Denmark, September 2000.

[Ryan,99] Ryan, N., "ConteXtML: Exchanging Contextual Information between a Mobile Client and the FieldNote Server", Project Homepage. http://www.cs.ukc.ac.uk/projects/mobicomp/fnc/ConteXtML.html

[Salutation,99] Salutation Consortium. "Salutation Architecture Specification (Part-1)", Technical Report Version 2.0c, 1999.

[Satyanarayanan,85] Satyanarayanan, M., Howard, J. H., Nichols, D. N., Sidebotham, R. N. Spector, A. Z. and West, M. J., "The ITC Distributed File System: Principles and Design", Proc. 10th Symposium on Operating System Principles (SOSP), Orcas Island, Washington, U.S., ACM Press, December 1985.

[Satyanarayanan,90] Satyanarayanan, M., Kistler, J. J., Kumar, P., Okasaki, M. E., Siegel, E. H. and Steere, D. C., "Coda: A Highly Available File System for a Distributed Workstation Environment", IEEE Transactions on Computers, Vol. 39 No. 4, Pages 447-459. April 1990.

[Scheirer,00] MIT Media Lab, Affective Computing Research Project. "Expression Glasses Home page", http://vismod.www.media.mit.edu/people/rise/jocelyn/tttpage.html

[Schilit,94a] Schilit, B. N. and Theimer, M., "Disseminating Active Map Information to Mobile Hosts", IEEE Networks, pages 22-32, October 1994.

[Schilit,94b] Schilit, B. N., Adams, N. I. and Want, R., "Context-aware Computing Applications", In the Proceedings of the 1st International Workshop on Mobile Computing Systems and Applications, pp. 85-90, Santa Cruz, CA, IEEE. December 8-9, 1994.

[Schilit,95] Schilit, W. N., "A System for Context-Aware Mobile Computing", Ph.D. Thesis, Columbia University, New York, 1995.

[Schmid,01] Schmid, S., Finney, J,. Wu, M., Friday, A., Scott, A. C., Shepherd, W. D., "An Access Control Architecture for Metropolitan Area Wireless Networks", in Proc. 8th International Workshop on Interactive Distributed Multimedia Systems (IDMS), September 4-7, 2001, Lancaster, UK.

[Schmidt,98] Schmidt, A., Beigl, M. and Gellersen, H. W., "There is More to Context than Location", In Interactive Applications of Mobile Computing, Rostock, Germany, 24-25, November 1998.

[Schmidt,98b] Schmidt, A., Aidoo, K. A., Takaluoma, A., Tuomela, U., Laerhoven, K. V. and Velde, W. V., "Advanced Interaction in Context", In the Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC '99), pp. 89-101, Karlsruhe, Germany, Springer-Verlag. September 27-29, 1999.

[Schmidt,00] Schmidt, A., Takaluoma, A. and Mntyjrvi, J., "Context-Aware Telephony over WAP", Springer-Verlag, London, Ltd., Personal Technologies, Volume 4, pages 225-229. Short paper presented at Handheld and Ubiquitous Computing (HUC2k), HP Labs, Bristol, UK, September 2000.

[Scourias,00] Scourias, J., "Overview of the Global System for Mobile Communications", http://ccnga.uwaterloo.ca/~jscouria/GSM/gsmreport.html.

[Shafer,00] Shafer, S., Brumitt, B., and Meyers, B. "The EasyLiving Intelligent Environment System", CHI Workshop on Research Directions in Situated Computing, April 2000.

[Siewiorek,98] Siewiorek D., Smailagic, A., Bass, L., Siegel, J., Martin, R. and Bennington, B., "Adtranz: A Mobile Computing System for Maintenance and Collaboration", Proc. Symposium on Wearable Computers.

[Sikkel,97] Sikkel, K., "A Group-based Authorization Model for Cooperative Systems", In: Proc. of ECSCW'97, Kluwer Academic Publishers, 345-360 (1997).

[Spectrix,99] Spectrix Corporation. "Wireless Networks for People in Motion", http://www.spectrixcorp.com/products/spectrixlite.html. 1999.

[Spiteri,98] Spiteri, M. and Bates, J., "An Architecture to Support Storage and Retrieval of Events", In IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware '98), Lake District, UK, September 15-18 1998.

[Sony,01] Sony Electronics Viao Homepage. http://www.sonystyle.com/vaio/

[Sun,99a] Sun Microsystems. HotJava HTML Component Product Homepage. http://www.java.sun.com/products/hotjava/

[Sun,99b] Sun Microsystems. Jini Architecture Specifcation - 1.0. Technical Report, Sun Microsystems, January 1999.

[Sun,99c] Sun Microsystems. Jini Web Site, 1999. http://www.sun.com/jini/.

[Sun,00d] Sun Microsystems. PersonalJava Application Environment, 2000. http://java.sun.com/products/personaljava.

[Swedberg,99] Swedberg, G., "Ericsson's Mobile Location Solution", Ericsson Review, (4):214-221, 1999.

[TIC,01] The Lancaster City Council Tourist Board Web Site. The Lancaster GUIDE System, http://www.lancaster.gov.uk/attractions/

[UPnP,00] Universal Plug and Play Forum, "Universal Plug and Play Web Site", 2000. http://www.upnp.org/.

[Voyager,01] Object Space Product Web Site, available online at http://www.objectspace.com/products/voyager/

[Wade,00] Wade, S., "An Investigation into the use of the Tuple Space Paradigm in Mobile Computing Environments", Ph.D. Thesis, Computing Department, Lancaster University, September 1999.

[Waldo,99] Waldo, J., "The Jini Architecture for Network-Centric Computing", Communications of the ACM, 42(7):76-82, 1999.

[Want,92] Want, R., Hopper, A., Falcao, V. and Gibbons, J., "The Active Badge Location System", ACM Transactions on Information Systems, 10(1):91-102, 1992.

[Want,95] Want, R., Schilit, B. N., Adams, N., Gold, R., Petersen, K., Greenberg, D., Ellis, J. and Weiser, M., "An Overview of the ParcTab Ubiquitous Computing Environment", IEEE Personal Communications, 2(6):28-43, 1995.

[Want,96] Want, R., Schilit, B. N., Adams, N. I., Gold, R., Petersen, K., Goldberg, D., Ellis, J. R. and Weiser, M., "The Parctab Ubiquitous Computing Experiment", In Tomaz Imielinski and Henry F Korth, editors, Mobile Computing, pages 45-101. Kluwer Academic Publishing, 1996.

[WAP,00a] Wireless Application Protocol Forum Ltd. WAP Forum Web Site, 2000. http://www.wapforum.org/.

[Ward,97] Ward, A., Jones, A. and Hopper, A., "A New Location Technique for the Active Office", IEEE Personal Communications, 4(5):42-47, 1997. MPEG Video available at ftp.uk.research.att.com/pub/videos/qsif-200/bat-qsif-200.mpg

[Weiser,91] Weiser, M., "The Computer for the 21st Century", Scientific American, 265(3):94-104, September 1991.

[Weiser,93] Weiser, M., "Some Computer Science Issues in Ubiquitous Computing", Communications of the ACM, 36(7):75-84, 1993.

[Xircom,01] Xircom Netwave Product Homepage. http://www.xircom.com/cda/page/0,1298,0-0-1_1-1629,00.html

Appendix A

Creating a User Profile

A.1 Introduction

When a tourist to the city of Lancaster first receives their GUIDE unit they are expected to perform a short configuration task in order to specify their context constraints, that is, their personal preferences. This configuration task needs to be straightforward since it is their first experience of the system, and therefore, a simple wizard style interface has been adopted in order to guide users through the process of creating a profile. Chapter three (see section 3.7.1) provided an overview of the user interface used to customise the GUIDE system, the following section shows the configuration process in more detail.

A.2 The GUIDE User Preferences Wizard

Before the user is asked to perform any tasks a short summary is presented (see figure A.1 below).

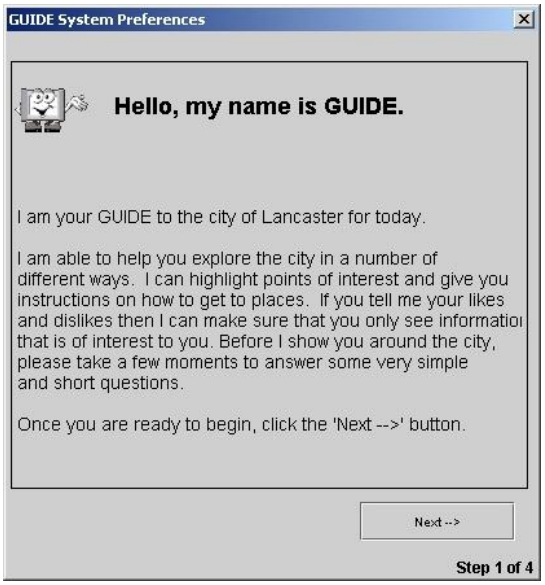

Figure A.1 - GUIDE Customisation wizard, step one

After clicking the next button, the following step (step 2) will be displayed, as shown in figure A.2. During this stage, the user is able to specify their name, age, preferred reading language and their specific tourist interests.

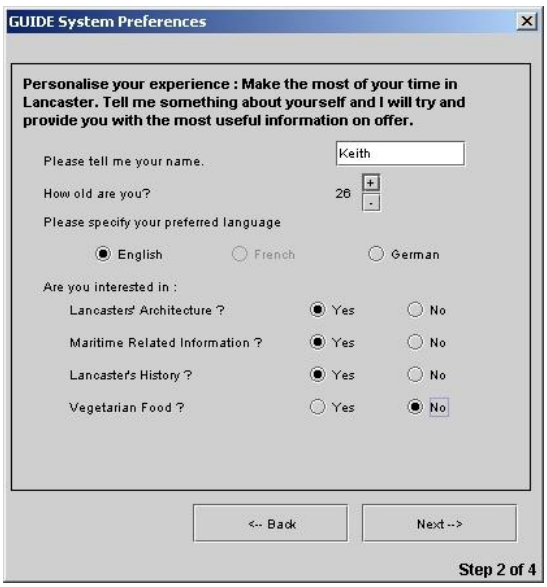

Figure A.2 - GUIDE Customisation wizard, step two

Once the user has chosen their personal details, they are ready to move onto step three to configure any privacy options relating to how their location information is shared and accessed by other group members. Step three allows a user to specify whether or not they with to be visible and/or contactable by others members of their group (if they are indeed travelling as part of a group), as shown in figure A.3.

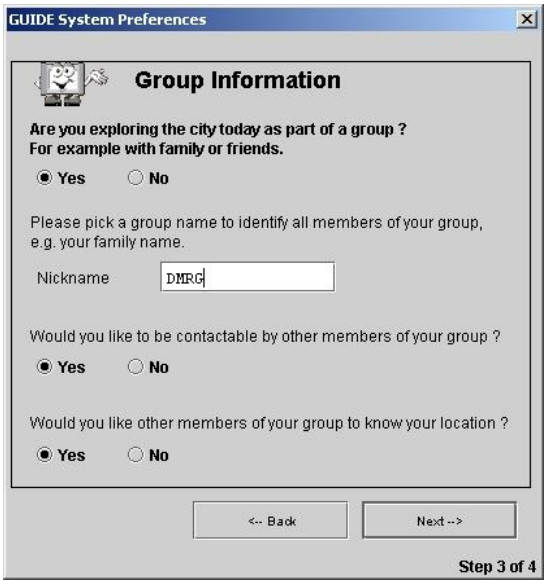

Figure A.3 - GUIDE Customisation wizard, step three

Once a visitor has chosen their privacy preferences at the group level, they are ready to move onto step four to configure any other context sharing properties. More specifically, step four enables a user to stipulate whether or not they wish to be visible and/or contactable by others members of the GUIDE system. Furthermore, a number of additional options are available which include whether they wish to have their real name revealed, choose to use an alternative nickname or remain anonymous, as shown in figure A.4.

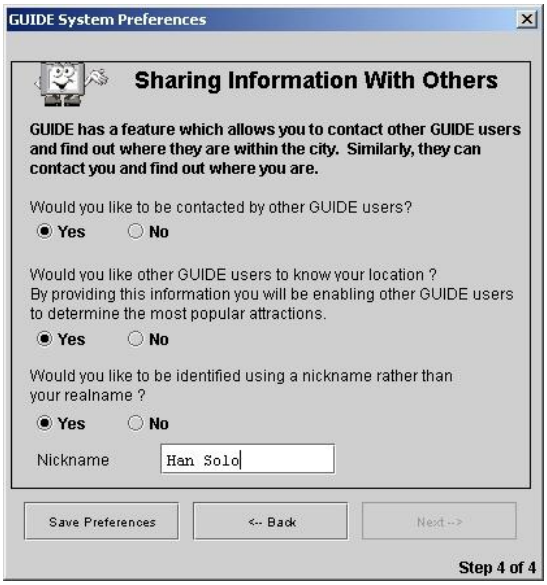

Figure A.4 - GUIDE Customisation wizard, step four

In addition to the four steps introduced above, an advanced option tab is also available to users wishing to further tailor the operating parameters. More specifically, the GUIDE application enables users to stipulate several QoS constraints associated with their location context. In essence, users are able to control several parameters in relation to making use of a location service: the frequency of updates required, the approximate battery consumption required and the positional accuracy required (as shown in figure A.5). It should be noted that a trade off exists between positional accuracy and power consumption. More specifically, a user who requests a higher degree of positional accuracy is assumed to require a location mechanism such as a local GPS compass. In a situation where positional accuracy is of less importance an alternative technology, such as the beacon based approach detailed in chapter three, may be utilised since this is likely to reduce the battery consumption of the mobile device.

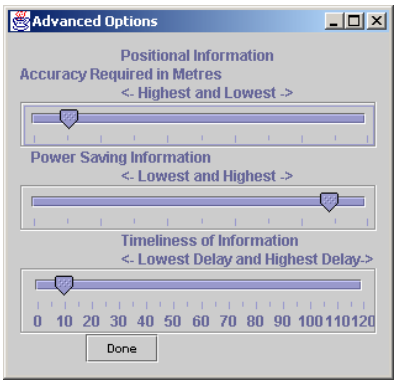

Figure A.5 – GUIDE Advanced Options

This simple configuration process provides enough information to enable the context-sensitive features found within the GUIDE system to be used effectively.

Appendix B

GUIDE Tags: Syntax and Semantics

B.1 Introduction

This appendix details the use of HTML tags embedded with hypertext pages to enable context sensitive content to be created dynamically within the GUIDE system. An overview of the tags can be found in section 3.4.3. This section describes in more detail the syntax and semantics relating to the range of tags in use by the GUIDE prototype application. Where appropriate, illustrative examples (including screenshots) are provided.

B.2 Syntax and Semantics

In order support the display of context-sensitive content to users, the GUIDE prototype application makes use of the following tags:

- **INSERT**: The insert tag is used to query both the information model (location objects) and user the model (user preferences) to determine contextual attributes to be inserted dynamically into hypertext pages. For example, to personalise a hypertext page and insert a visitor's name.
- **INTEREST**: The interest tag is used to highlight a particular section of hypertext as containing (potentially useful) information to users. This could

be historic or architectural information and in order to ascertain whether or not the information should be displayed the user model will be queried.

- **METATAG**: The meta tag is similar to the interest tag, however, the use of a meta tag does not involve any content to be modified dynamically. Instead, the tag is used simply to denote that a particular page contains tourist specific information. For example, `<GUIDETAG METATAG HISTORY>` would be used to identify a page containing historical information relating to one of the city's landmarks.
- **COMMENT**: The comment tag is used to query the information model (location objects) to determine whether or not any previous users of the system have left any personal comments that can be retrieved. In essence this provides a virtual guest book to users of the GUIDE system.
- **COLLABORATE**: The collaborate tag is used to query the GUIDE cell servers for a list of users present (currently or previously that day) at a particular location. For example, a user wishing to visit Lancaster Castle may decide to ask another user's opinion on the landmark before making the journey across the city. The collaborate tag would enable the system to reveal others currently at that location taking into account their privacy concerns (see section A.2).
- **UPDATE**: The update tag is used to update state within both the information and user models.

Figure B.1 provides an overview of the steps involved in producing a hypertext page dynamically. The following sub-sections describe, using examples, the GUIDE tags currently in use by the current GUIDE prototype application.

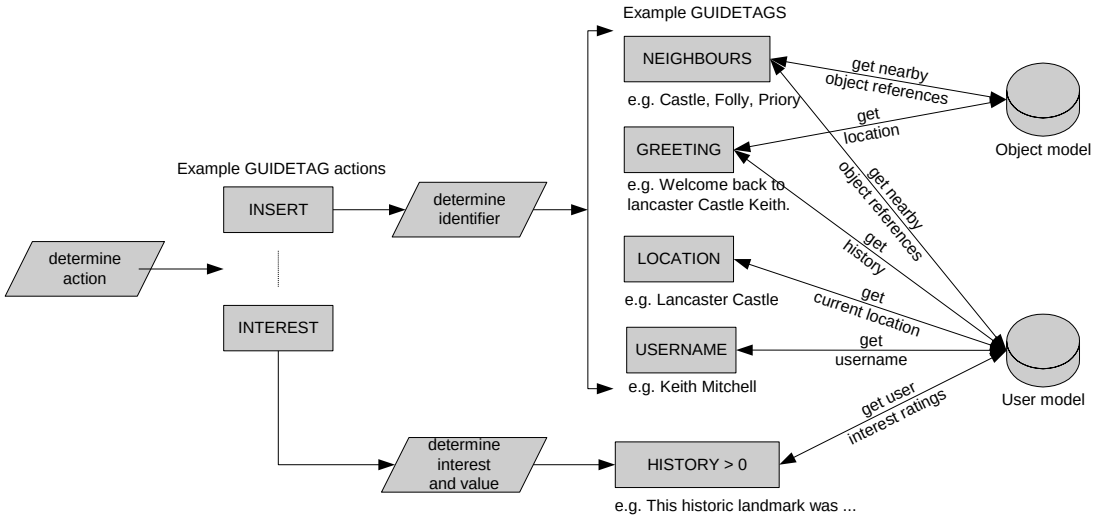

Figure B.1 - Creating a dynamic hypertext page

B.2.1 The INSERT Tag

The insert tag is used to query both the information model (location objects) and user the model (user preferences) to determine context dynamically. For example, to personalise a hypertext page and welcome a particular user to a landmark, a combination of tags can be used, as shown in figure B.1.

When the GUIDE filter processes an `INSERT` tag, any aspect of the information or user model may be queried and the result (which is always hypertext markup) forms part of the personalised page. For example, the following tag; `<GUIDETAG INSERT FULLUSERNAME>` would result in the following being displayed to a user (see figure B.2

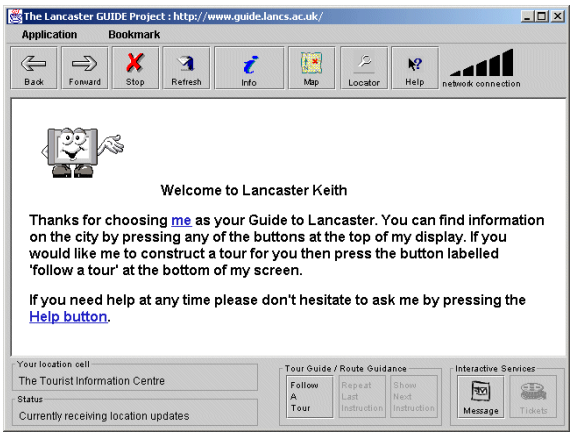

Figure B.2 - Personalising a tourist page

Inserting information relating to nearby locations involves both the user model and environment model. Figure B.3 illustrates the process involved when the GUIDE filter processes a hypertext page containing the `<GUIDETAG INSERT NEIGBOURS>`.

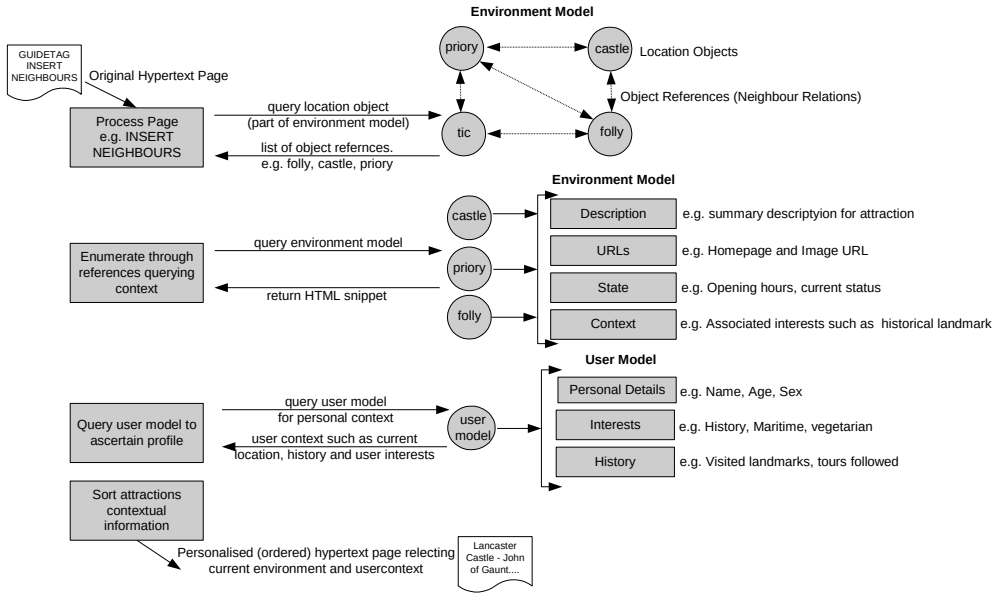

Figure B.3 - Inserting nearby places

Figure B.4 shows the result viewed by a user after querying the system for a list of nearby attractions.

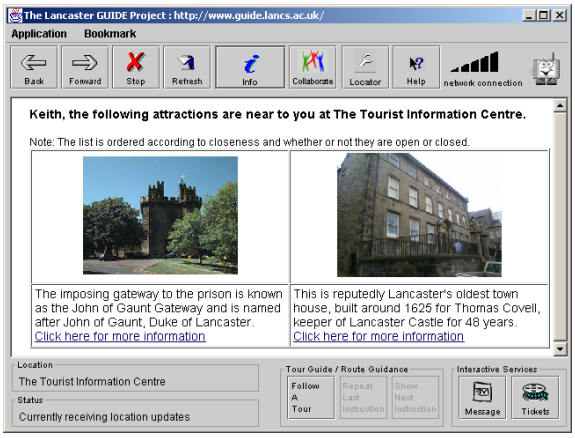

Figure B.4 - Creating dynamic hypertext content

B.2.2 INTEREST

The interest tag is used to highlight information with particular attributes, for example, tagging some hypertext content as historical. The use of this tag requires the

use of a </GUIDETAG> closing tag. The processing of this tag involves querying the context associated with the user profile in order to determine their user preferences. If a query evaluates to true the hypertext information contained within the tag is displayed, otherwise it is not. The statements can be combined using the AND and OR logical operators. The syntax is as follows;

```
<GUIDETAG FUNCTION  ( STATEMENT  AND | OR STATEMENT ) >
```

and examples include

```
<GUIDETAG INTEREST (HISTORY GREATER 50) AND (ARCHITECTURE LESS
50) >
      … HTML …
</GUIDETAG>
```

B.2.3 METATAG

The meta tag is used to highlight hypertext pages with particular interests. These tags cause no visible change to the page content but indicates that a hypertext page contains content of particular relevance. The use of this tag causes the user preferences to be queried, for example <GUIDETAG METATAG HISTORY>, would be used to indicate that the hypertext pages contains historical content. If the user profile is queried and it is shown that the user has no interest in this option then a virtual counter relating to the history preference will be incremented. Should the virtual counter reach a specified limit, the system will notify the user and bring to their attention that they have visited a large number of pages containing history related information (see figure B.5 below). This tag is used to ensure visitors to not miss large amounts of hypertext information relating to a specific subject.

Figure B.5 – Updating the user profile

B.2.4 COMMENT

The comment tag is used to query the information model (location objects) to determine whether any previous users of the system have left personal comments relating to a particular landmark. This feature is used to allow visitors to query the system and search for comments/feedback relating to particular attractions, thus acting as a virtual guest book to the city's landmarks. Furthermore, any user of the system is able to leave their own comment relating to any of the city's attractions. For example, the following tag `<GUIDETAG COMMENT SHIRE>` would result in the following screen being displayed to a user of the system (see figure B.6).

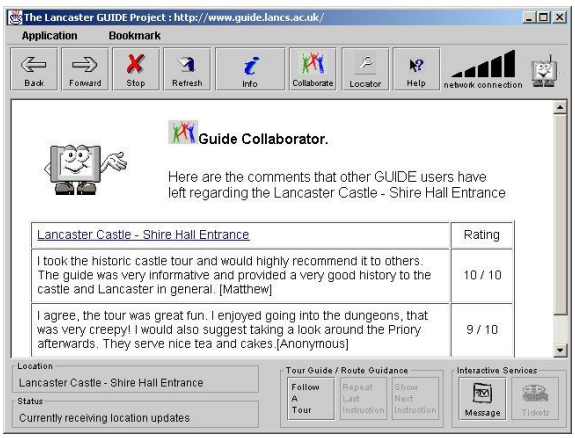

Figure B.6 - Viewing comments left by other tourists

B.2.5 COLLABORATE

The collaborate tag is used to query the GUIDE cell servers for a list of current or pervious users (that day) that have been shown to have visited a particular location.

For example, a visitor may wish to have lunch at the Folly Café and may be interested in gaining an opinion of the café from another visitor before entering the café. The collaborate tag would enable the system to reveal others currently at that location taking into account their privacy issues (see Appendix A). The GUIDE filter object expands the tag by checking that the visitor has network connectivity and (given that this connectivity is available) makes a remote request for a list of GUIDE users currently in the specified cell. The server responds to the query by returning the identity of those visitors currently recorded as being present at the specified location. For privacy purposes, the actual name of a visitor is only passed back to the client if the visitor has agreed to be contactable (and the permissions for this are stipulated in the user's profile). If a visitor has requested anonymity then a unique id is assigned and returned for that visitor in order to mask the visitor's identity, e.g. anonymous1. For example, the following tag `<GUIDETAG COLLABORATE FOLLY>`, would result in the following page being displayed to a user of the system (see figure B.7 below).

Figure B.7 - The use of the collaborate tag

After viewing this page, a user wishing to initiate communications with a fellow GUIDE user to ask their opinion of an attraction may request them to provide a simple rating. To do this the following dialog box can be completed, as shown in figure B.8.

Figure B.8 - Requesting a rating for a city attraction

B.2.6 UPDATE

The update tag is used primarily as part of the locator when a user becomes lost. When visitors leave cell coverage and up-to-date positioning information becomes unavailable, the GUIDE system tries to locate the visitor by establishing a form of partnership between itself and the visitor. More specifically, the visitor is shown a series of thumbnail pictures showing attractions in the vicinity of the visitor's last known location, as shown in figure B.10. Hopefully, the visitor is then able to recognise and select one of the pictures in order to enable the GUIDE system to once again ascertain the visitor's location within the city. Once the location is determined an `UPDATE POSITION` tag within a hypertext page can be used to force the system to update it's location context (i.e. a virtual beacon) and user interface (see figure B.9, B.10 and B11).

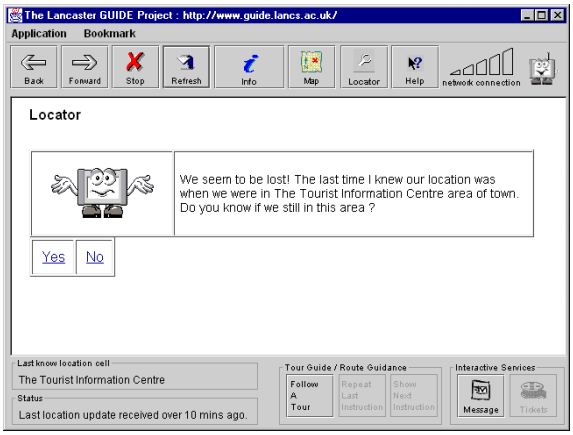

Figure B.9 - The GUIDE system detects that a user is lost

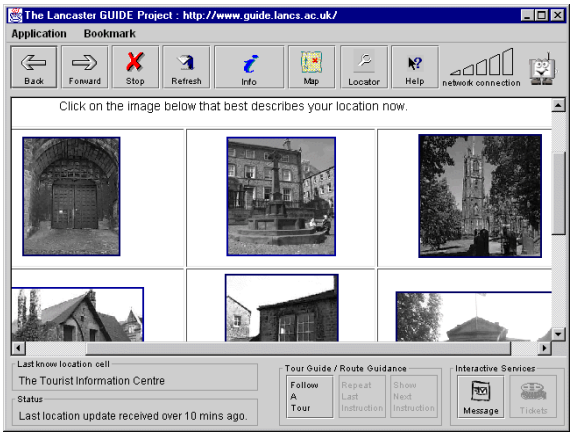

Figure B.10 - The GUIDE system presents a series of thumbnails

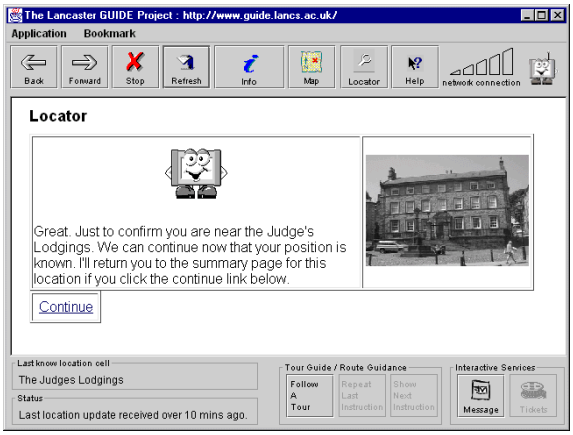

Figure B.11 - The GUIDE System updates the user interface

Appendix C

Developing a Context-Sensitive City Tour

C.1 Introduction

This appendix provides a pseudo-code algorithm and flowchart to detail the tour creation process used by the Lancaster GUIDE system when creating personalised, intelligent tours of the city of Lancaster. The creation of a city tour does not involve determining the shortest route between attractions within the city, but offers a more intelligent tour based on a variety of contextual factors such as visitor interests, the attractions of interest, travel constraints (such as wheelchair use), time (e.g. time available to take a tour), time sensitivity of attractions (since in many cases there are good and bad times to visit specific attractions), weather, and financial constraints.

In essence, when constructing a guided tour of the city, the tour guide component attempts to quantify (evaluate) the quality of a tour by allocating numeric values (scores) to attractions and routes between attractions. These values are influenced by both the current context (weather, time etc.) and the user's preferences and tours can be ordered by comparing their total scores. In more detail, the system dynamically generates a full set of tours that encompass the attractions requested, produces a total

score for each permutation and then recommends to the visitor the tour with the highest score (most suitable).

Furthermore, the system incorporates a number of "tricks" to try and improve a given tour's overall score. For example, consider the following scenario. It is 9.30am and a tourist has just picked up a GUIDE unit from the TIC. They ask the system to generate a day-long tour that takes in just two places, a nearby museum (which opens at 10.30 am) and a park on the other side of town. The system recognizes that the best order is to visit the museum first and then spend the afternoon in the park and will suggest "padding" activities (e.g. nearby cafés, exhibitions or related landmarks) to occupy the visitor until the museum opens. The pseudo-code algorithm shown in figure C.1 forms the basis of the tour creation process. Figure C.2 shows a flow diagram outlining the tour creation process.

```
find all permutations of required places

for each permutation
       averageTime = total travel time + average duration of each place + if (user wants lunch) 1 hour time for lunch
       minTime = total travel time + min duration of each place + if (user wants lunch) 1 hour time for lunch

       if (averageTime > time available)
                if (total travel time + min duration of each place > time available)
                    delete this permutation
                else
                    mark this permutation bad choice

        if (no permutations left)
            ask user to remove a required place

        while (no tour)

            possible invalids = 0

            for each valid and not bad choice permutation(using minimum duration)
                total score = 0
                for each place in current permutation
                    if (12 < time < 14 and user wants lunch)
                        find food place in nearby places with traveltime from last place to there and from there to next place
                            + averageDuration there fits in 1 hour and its score != -1
                        insert this place in permutation
                        add 1 hour to time
                        add score to total score
                        for each 5 min later than 13.30h substract 5 from total score
                    time = time + travelTime from last place to this
                    score = place.evaluate(time , budget)
                    if (score == -1)
                        invalids = invalids + 1
                        if (possible invalids < invalids)
                            mark permutation invalid
                        else
```

```
                    add 30 min to time and evaluate until score != -1
                    if (never valid)
                        mark permutation invalid
                    else
                        find place in nearby places with
                            traveltime from last place to there and from there to next place
                            + averageDuration there fits in timegap and its score != -1
                            insert this place in permutation
            else
                add score to total score
    if (all tours invalid)
        do the above with bad choices

    if (still all tours invalid)
        possible invalids + 1
        if (possible invalids > number of required places)
            return no tour
        else
            if (valid tours)
                tour found
                return tour with highest total score
```

Figure C.1 - The tour creation process algorithm

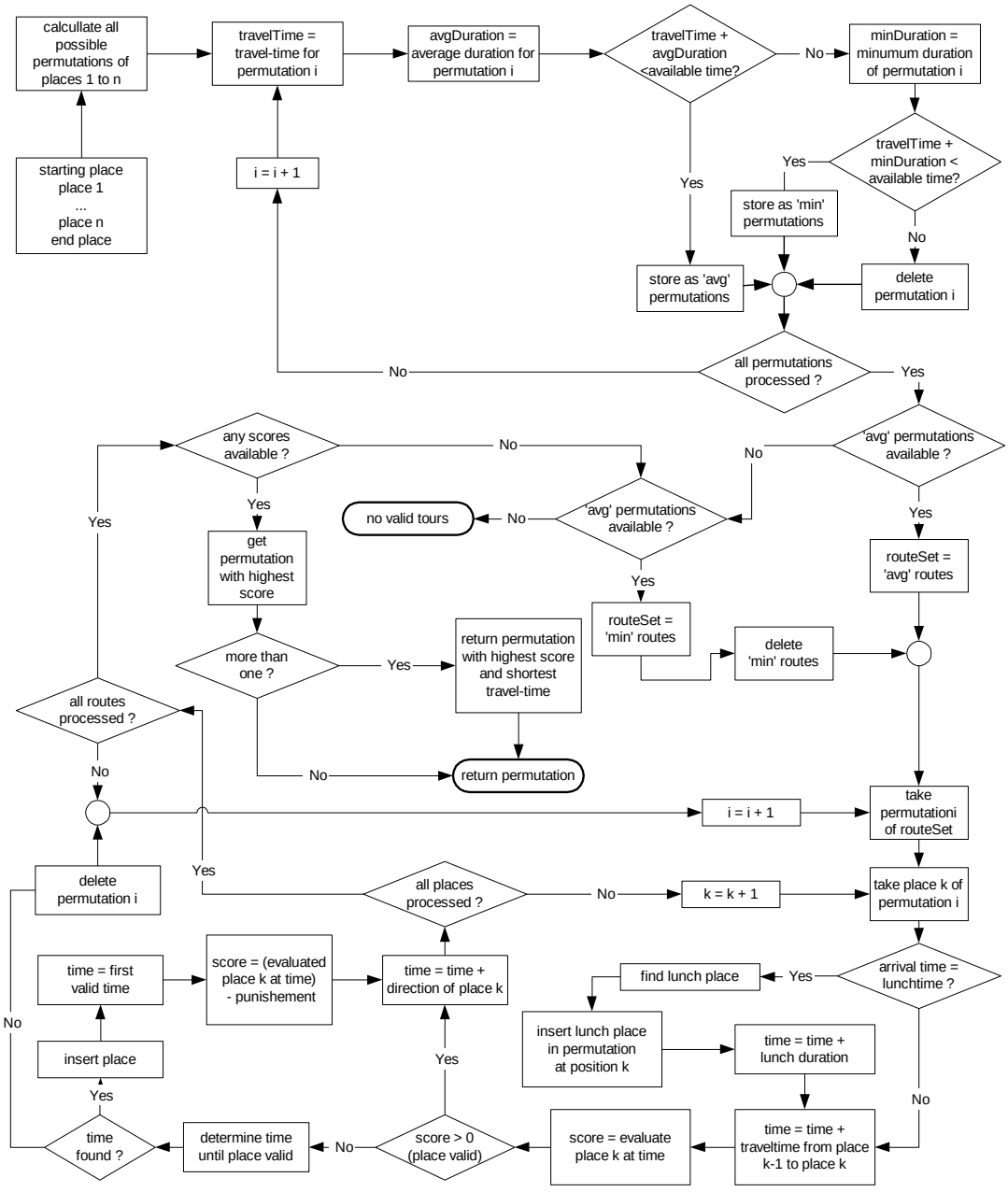

Figure C.2 - GUIDE tour creation flowchart

Appendix D

An XML User Profile

D.1 Introduction

This appendix provides a sample XML description relating to the user profiles used within the GUIDE application prototypes. A series of Java based XML parsers are utilised to interpret such XML documents in order for instances of the `GuideUserProfile` class to be created. Furthermore, serialised instances of a `GuideUserProfile` object stored within the context repository may be un-serialised and stored as an XML document, enabling access to the data devices such as WAP enabled mobile phones.

D.2 A User Profile Specified in XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<profile>
<applications>
 <application>
    <applicationname>GUIDE</applicationname>
  <username>km</username>
  <fullname>Keith Mitchell</fullname>
  <age>26</age>
  <anonymous>false</anonymous>
  <contactable>true</contactable>
  <language>english</language>
<interests>
        <history>100</history>
        <maritime>100</maritime>
        <vegetarian>0</vegetarian>
        <architecture>100</architecture>
</interests>
<contexttypes>
<context>
        <type>location</type>
        <constraints>*</constraints>
        <scope>all</scope>
</context>
<context>
        <type>user</type>
        <constraints>*</constraints>
</context>
</contexttypes>
</application>
</applications>
</profile>
```