Exploiting space and location as a design framework for interactive mobile systems

Alan Dix[†], Tom Rodden[‡], Nigel Davies[‡],
Jonathan Trevor[‡], Adrian Friday[‡], Kevin Palfreyman[‡]

[†] aQtive limited
Birmingham Research Park
Birmingham, B15 2SQ UK
and Staffordshire University, UK

[‡] Department of Computing
Lancaster University
Lancaster, LA1 4YR, UK

Abstract

This paper considers the importance of context in mobile systems. It considers a range of context issues and focus on location as a key issue for mobile systems. A design framework is described consisting of taxonomies of location, mobility, population and device awareness. The design framework informs the construction of a semantic model of space for mobile systems. The semantic model is reflected in a computational model built on a distributed platform that allows contextual information to be shared across a number of mobile devices. The framework supports the design of interactive mobile systems while the platform supports their rapid development.

Categories and subject descriptors

H.1.2 [**Information Systems**] User/Machine Systems , C.2.4 [**Computer Systems Organization**] Distributed Systems - Distributed applications; H.4.3 [**Information Systems**] Communications Applications , H.5.3 [**Information Systems**]Group and Organization Interfaces - Synchronous interaction; Theory and Models

General terms

Design , Human factors, Theory

Keywords and phrases

Mobile Systems, Context Information, Virtual Space, Awareness, Platform Support, Shared Interaction, Design Framework, Location sensitive applications

Primary Contact

Tom Rodden
Department of Computing, Lancaster University
Lancaster, LA1 4YR, , UK
Tel: +44 1524 593823
Fax: +44 1524 593608
Email: tom@comp.lancs.ac.uk

1. Introduction

1.1 The emergence of connected mobility

The increased availability of communication facilities has seen a shift in the nature of mobile computers systems and applications. Prior to the widespread emergence of mobile communication services the work on mobile computer systems has tended to focus upon the development of small devices and the forms of interaction that they afford. In particular, we have seen considerable focus on the development of techniques and approaches to overcome display limitations and the ergonomics of new interactive devices [Bass, 1997]. However, the development of a new class of mobile devices that link with telecommunication services to offer connections to other systems mean that we need to extend our consideration of design for mobile systems. These mobile systems are particularly problematic because of the various ways in which they break assumptions that are implicit in the design of fixed-location computer applications, leading to new design challenges for human–computer interaction.

One immediate issue arising from the nature of wireless communications are delays and outages, leading to slow and unpredictable temporal characteristics at the user interface. These issues have already been addressed in some detail in our previous work on pace of interaction and practical experience in building collaborative mobile applications [Dix, 1992; Davies 1994; Dix, 1995]. They are also considered in the wider study of temporal issues in HCI [Johnson, 1996; Johnson, 1997; BCSHCI, 1997; Howard and Fabre, 1998].

Another issue that has been of considerable importance is the development of context sensitive devices that have a distinct awareness of their location and other devices [Davies 1994, Fickas, Long 996, Want 1995]. However, the treatment of these issues has, so far, been predominantly focused on the design of specific devices and applications. In this paper we are aiming to produce a broader view of context and location-aware computation. In particular, we will present a framework to support the design of interactive mobile systems based on an understanding of location within these systems.

1.2 Context and mobility

In mobile systems, as in other areas of human–computer interaction, it is not sufficient to focus on the specific interface of a device. The device operates within a broader context. This context includes the network and computational infrastructure, the broader computational system, the application domain and the physical environment.

Each of the different contexts represents a different part of the design space within which mobile systems must be placed and the features of infrastructure, system, domain and environment all suggest trade-offs that developers must address in realising mobile interactive systems. Currently, designers undertake this trade-off with little support or guidance as little is known about the design space into which mobile applications are placed.

The design framework developed in this paper focuses on the context sensitive nature of mobile devices in the design and development of cooperative mobile systems. We wish to chart the design space for mobile systems to allow developers to consider the properties of mobile systems under construction and how they may be related to other applications and systems. In developing this design framework we focus particularly on location, as ideas of space and location are of paramount importance in any consideration of the context of these systems.

In undertaking this task we wish to build upon the research lessons from research on temporal issues in interaction. This research community has successfully created frameworks to improve the general understanding of temporal aspects prior to design and construction of new forms of technology. We wish to apply a similar tactic by moving from a theoretical consideration of the context of mobile systems, through an examination of location and the development of a framework for the design of mobile systems, to the development of models to support these systems. Our particular interest is in the way in which the interactive nature of cooperative mobile systems drives the development of supporting infrastructures, and our endpoint is a computational infrastructure we have developed for these systems.

1.3 Structure of this paper

In developing our design framework we particularly focus on the importance of location and space in mobile systems. However, it would be unwise for us to suggest that location is the only manifestation of context in mobile systems. In order to place our framework within the broader picture of the design of interactive mobile applications we begin, in section 2, by considering the nature of the context in which interaction with mobile applications takes place. This broader consideration of context acts as a general backdrop for our focus on location as a means of designing for context sensitive mobile systems.

In section 3, we address the need to consider location in terms both of the physical space in which a mobile device exists and of the virtual models of space exploited by applications. The importance of location underpins the development, in section 4, of multiple taxonomies of location, mobility, population and device awareness. This conceptual analysis allows us to exploit location as a means of understanding interactive mobile applications. In section 5, we build upon these taxonomies to suggest a simple semantic model of space that allows us to more generally represent and reason about the location of devices. We then use this semantic model to develop a simple computational model and supporting infrastructure that allows the understanding of location to be conveyed across a number of devices working in tandem to realise cooperative mobile applications.

Our computational model of location is realised on top of a distributed architecture and infrastructure. In section 6, we discuss this architecture and also general architectural issues for contextual systems. This section ends with a short description of the architecture of *onCue*, a commercial system built using many of the principles espoused in this paper.

2. The contextual nature of mobile systems

In considering the design of mobile systems we wish to focus particularly on the situation where mobile devices behave differently and offer different interaction possibilities depending on the particular context in which the system is being used. For example, in the development of mobile multimedia guides, such as the systems at Georgia Tech [Long, 1996] and the Lancaster GUIDE [Davies, 1998], the information presented to the user and the interaction possibilities are strongly linked to the location where the device is being used. In these cases interaction is no longer solely a property of the device but rather of the *device in context*. While location is often the principal determinant used to represent this context and the main focus of this paper it is worth briefly considering the general importance of context in mobile systems and why understanding and modelling this context is important.

A considerable amount of research surrounding the development of mobile devices has obviously focused on the portable nature of these devices and the technical problems of implementation. Mobile computing devices represent real technical challenges and have always stretched the state of the art in terms of displays and interaction devices.

The emergence of mobile telecommunication standards such as GSM and the increased availability of these services have also led more recently to the development of devices that provide mobile access to on-line services (e.g. the Nokia communicator). This merging of computer and communication facilities allows the development of systems that provide immediate on-line access to information. These portable networked devices have also been combined with the use of GPS technologies to develop portable devices that are aware of their position [Long, 1996].

The current generation of portable devices have an awareness of their setting and an increased ability to access network resources. This means that we need to balance the current consideration of the interaction properties of individual devices with a broader consideration of the context of use. The importance of context to interactive systems is not unique to mobile devices and is already reflected in research in ubiquitous computing, wearable computers and augmented reality [Weiser, 1991, 1994; Aliaga, 1997] and more recent work at MIT on the development of devices that exploit context to provide an ambient awareness of interaction [Ishii, 1997].

The term 'context' has become problematic for interactive systems development and is itself the subject of some debate. One aim of the growing focus on context is to allow the highly situated nature of interactive devices to be reflected in the design of systems. This focus on the situated nature of these devices reflects their growing acceptance and the need to allow them to closely mesh with existing practices, and mirrors previous work in the development of interactive systems within CSCW [Hughes, 1994]. Rather than engage in this much broader discussion we wish to concentrate on location. However, in the remainder of this section we wish to briefly characterise some of the broader debate on context that is relevant to mobile systems.

2.1 Context in the design of mobile systems

In order to reflect the broader role of context in our consideration of location we wish to unpack what we might mean by the term context and how we may exploit it to determine different interaction possibilities for mobile systems. In following sections we consider some of the ways in which context has played a key design role in the development of distributed mobile applications. We outline some of the different forms of context that influence interaction with mobile systems before we consider in central role of location in section 3. Our consideration of context moves from the nature of the underlying *infrastructure context* to consider, the overall *system context*, the broader application *domain context* and finally the actual *physical context*.

Infrastructure context

The interaction offered by mobile applications is not solely dependent on the particular features of the mobile devices used. Rather it is a product of the device and the supporting

infrastructure used to realise the application. The impact of the properties of the supporting distribution infrastructure on different styles of interaction has been discussed in CSCW and HCI [Greenberg, 1994]. In mobile systems the nature of the infrastructure is even more likely to change as the application is used, and the sort of service available may alter dramatically. This variability in the infrastructure can dramatically affect interaction and it is essential that interaction styles and interfaces also reflect the state of the infrastructure.

In essence, the user interfaces to mobile applications must be designed to cope with the level of uncertainty that is inevitably introduced into any system that uses wireless communications. Consider our experiences in the development of an advanced mobile application used to support collaborative access to safety critical information by a group of field engineers [Davies, 1994]. If one of these engineers becomes disconnected from the group as a result of communications failure then it is vital that the remaining users' interfaces reflect this fact. For example, if an engineer is about to work on a cable it is important that the system either (a) correctly reflects the current state of the cable or (b) clearly shows that the information is not current. If this does not hold the engineer could easily touch a live cable with potentially fatal consequences. Reflecting this information requires interaction between the application's user interface and the underlying infrastructure via which failures will be reported. In addition, if the information being manipulated is replicated by the distributed systems platform the validity of each replica will clearly be important to the engineers. In this case the user interface needs to reflect this platform information.

System context

Most advanced mobile applications are distributed in nature. Rather than functionality residing solely within a single machine (or device) it is spread across the system as a whole. This means we need to consider the interactive properties of the system in terms of the distributed nature of the application. This is particular true when we consider issues of pace and interaction [Dix, 1992]. For example, rapid feedback is an accepted premise of HCI design and many applications provide direct manipulation interfaces that rely on rapid feedback. The development of distributed applications and the impact of the delays inherent in the technical infrastructure has already seen a reconsideration of feedback [Dix, 1995, Ramduny and Dix, 1997]. The need to consider the overall functionality of the application and to design structures that provide appropriate access to different levels of functionality is amplified in the case of mobile applications where the infrastructure may vary dynamically as the application is in use.

Consider, for example, the development of caching strategies for field engineers who will only ever be examining or servicing units within a sub-region of a particular area. The choice of the appropriate location to cache information will depend on the required feedback, the safety-criticality of the information, the speed and reliability of different parts of the network, and on who else is likely to be using and updating the cached information. A local writeable cache would improve the feedback for an individual user. However, it may also make it appear that the user's data changes have been reflected in the system as a whole, when, in fact, the connection between the cache server and the data server is broken and other users are seeing dangerously out-of-date information.

Another aspect of this system context is the extent to which a device is aware of other devices

in its vicinity and, related to this, the extent to which an application is aware of other applications. This is important, partly because such devices can adversely affect one another as they contend for resources, but more importantly, because combinations of devices may be able to offer more advanced services to the user. This can lead to a planned or accidental emergent behaviour of the devices as a group, which is not defined in any individual device. One example of this is onCue (see Section 6.3), which only offers certain services when particular software is available on the local machine.

Domain context

As well as addressing the infrastructure and system issues discussed above, distributed mobile applications need to consider the semantics of the application domain. The situated nature of advanced multimedia applications is such that design needs to explicitly identify the nature of the work being supported and the practicalities of this work. In doing so, developers need to consider the relationship between the mobile devices and their users and how this can be used to determine the nature of the interfaces presented. In the case of mobile applications the normal design considerations are amplified by the need to consider the limited interaction facilities of mobile devices.

Mobile devices are intended to be readily available and useful to the community of users being supported. As a consequence we need to consider the highly situated nature of this interaction. Developing a clear understanding of what people do in practice and the relationship with technology is essential to informing the development of these applications. The relationship between users and mobile technology is still unclear and few studies have taken place that consider the development of mobile cooperative applications [Davies, 1994].

For example, we may choose to exploit the personal nature of these devices to associate mobile devices with users. This allows us to tailor applications to allow them to be sensitive to the identity of the user of the device. This information may be exploited along with additional contextual information (e.g. location) to present appropriate information. One example of this would be a particular doctor visiting patients within a hospital. At a particular bed, who the doctor is and their relationship to the patient in the bed may determine the information presented. Contrast this situation with the development of a museum guide where the devices need to be considered as general purpose and no information is available about the relationship between users and the artefact being described.

Another aspect of the domain is the level of trust and mutual awareness between participants in collaborative interactions. This is particularly important if devices are to be used to identify users and potentially make information about their location and what they are doing available to others. In this case a consideration of the issues of privacy and the need for some management of privacy is essential [Harper, 1992].

Physical context

Finally, mobile computer systems are likely to be aware of, or embedded into, their physical surroundings. Often this is because they are embedded in an application-specific device, for example in a mobile phone or car. In these situations the computer system is mobile by virtue of being part of a larger mobile artefact. This context can and does affect the application

interface: for example, the telephone directory within a mobile phone can be very different from one in an independent PDA. Another example is a car radio (now often computer controlled) which has different design considerations to a static radio including the need to automatically retune as the car travels between local radio areas and transmitter zones. Because the computer systems are embedded into application-specific devices they may also be aware of their *environmental context*, for example the speed of the car. Some of this sensory information may be used simply to deliver information directly to the user, but some may be used to modify interface behaviour. For example, in a tourist guide, increasing text size in poor lighting conditions, or, in a car system, limiting unimportant feedback during periods of rapid manoeuvring.

Context in context

Context is largely about relationship, the four forms of context consider in this section have focused on the relationship between an interaction device and surrounding elements. These forms of context, the associated relationships and the issues raised are summarise in figure 1.

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 feedthrough, 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 dependent information, use of environmental sensors

Figure 1. Taxonomy of context

Although each of the different kinds of context discussed in this section are worthy of further study in their own right, this paper is predominantly concerned with physical context and the use of location in determining this. In the following section we discuss the central role of location and physical context for mobile systems. This is partly because of our desire to produce a computational infrastructure to support location-aware applications but also reflects the dynamic changes of location as a unique feature of mobile systems. However, as you would expect, these various forms of context are closely related and we will see elements of various other kinds of context throughout this paper.

3. The importance of location in understanding context

Clearly, the very idea of 'mobility' demands an understanding of location and one of the unique aspects of mobile devices is that they can have an awareness of the location within which they are being used. Furthermore, this location information may be exploited as a means of understanding the overall context within which the system is placed. Essentially, location becomes a useful indexing device from which to infer the overall context influencing

the mobile application: in order to ask "what devices are near this device" (system context) we need to know the location of this device and others; it only makes sense to measure the environment (physical context) when a device is physically located in space. Furthermore, the need to be contextually aware in other ways depends to a large extent on the mobility in space of devices. If devices are spatially static, many aspects of their environment are also static, or at most slowly varying.

3.1 Location and space

Any notion of location puts the device within some form of space. The space within which a device is located may also contain other devices and users with which the device may interact. A device involved in a mobile system can be considered as

- having location in the space
- having an effect on the space (and devices and user within it)
- being subject to influencing events from the space (and devices and users within it)

Essentially, devices are situated and embedded within a space and their interaction is mediated through this space (figure 2). Consequently, understanding the nature of their location in that space is key to understanding the nature of the mobile system being designed and provides a means of reflection on the context.

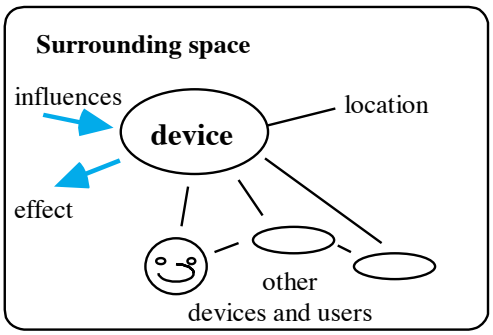

Figure 2. A device situated in space

If the device in figure 2 and other devices were at fixed locations, then the nature of these interactions would be one of configuration. However, the interesting and challenging nature of mobile interfaces is the changing nature of these relationships. So, to have an overall model of spatially situated interaction, we need to understand:

- location in space (of the device and other bodies)
- mobility through space (of these)
- the kinds of bodies populating the space (which the device may interact with)
- the awareness (of the device) of these other bodies

In section 4, we will develop of each of these but before we consider the different relationships between a device and space it is worth reflecting on what we may mean by space.

3.2 Of real and virtual worlds

Focusing on a device situated in space is only the start of any consideration of the importance of space to mobile systems. Consider a purely physical device like a lawnmower: it inhabits the real world and can be used to affect the real world. *It exists only in a single space* and its relationship is only with the physical world it inhabits. Its location is unique to that space and its influences and effects are only through the physical space within which it resides.

However, this is not the case for many mobile systems and our interaction with mobile systems. Computers and mobile devices are different in that we can consider their existence and presence in terms of many spaces. They can be though of as simultaneously inhabiting a real world and some form of virtual world (or indeed multiple virtual worlds). A computational device's ability to exist in the physical world while also having an existence in an electronic or virtual worlds is significant to any consideration of mobility.

The emergence of virtual space

Our consideration of an existence in an electronic world extends beyond the realm of virtual reality and we would suggest is equally evident as we surf the web, use ftp to access remote files or even simply explore our own file system. In all of these cases we are in a sense inhabiting virtual space. Even the vocabulary we use reflects this: we 'visit', 'explore', 'go to', 'navigate' ... our web browsers even have a button to go 'back'. This turning to virtual spaces and spatial approaches generally grows from the use of spatial metaphors and techniques to represent information and action in electronic systems. One of the early examples of the use of spatial metaphors includes the use of a rooms metaphor to allow the presentation of information [Henderson, 1985]. From these early spatial approaches we have seen concepts of spatial arrangement exploited in the development of desktop conferencing systems such as Cruiser [Root, 1988] and more generally in the work of Mediaspaces [Gaver, 1992].

The recent development of cooperative systems in CSCW has also seen a growing application of concepts drawn from spatial arrangements. These include the development of *groupkit* to form *teamrooms* [Roseman, 1996], the emergence of the *worlds* system [Fitzpatrick, 1996] and the use of a notion of places to support infrastructure [Patterson, 1996]. This exploitation of virtual spaces is most notable in the development of shared social worlds existing solely within the machine [Benford, 1995]. However, the use of space and virtual spaces has not been isolated to an existence solely within the computer and a number of researchers have considered how space and location can be considered both virtually and physically within the development of applications. This is most evident in the augmenting of existing physical spaces to form digital spaces populated by electronically sensitive physical artefacts (or tangible bits) [Ishii, 1997] that are sensitive to their position within both physical and virtual space.

Combining the real and the virtual

The work in tangible bits undertaken by Ishii [1997] represents the start of a trend to interweave real and virtual spaces. This work exploits the combined use of a number of devices within a space so that their physical manipulation can be used to generate a computational (or virtual) effect. Various other strands of recent research have explored

these boundaries between the physical and virtual including wearable computing and augmented reality. One example of this has been the explicit development of boundaries that span between the physical and the virtual [Benford, 1998]. We would suggest that this interplay between the real and the virtual is at the core of the design of cooperative mobile applications, as devices and users have a location and presence that is both virtual and physical and each is available to the computer application.

This interplay between the real and the virtual provides a starting point for the development of our taxonomy. A direct result of the need to recognise this coupling is that many of the categories we will consider for understanding mobile and context-aware computation have counterparts in both the real physical world and the virtual electronic world. There are important differences - the virtual world does not always behave in ways we have come to expect from the physical world - and designers and developers often exploit these differences.

In particular, even the object of interest for mobile computation may have a physical or virtual existence depending on the nature of the application. At one extreme we have hand-held GPS systems that simply tell you where you are in physical space – perhaps these do not even rank as mobile computation. At the other extreme there are agents that only have an existence within the virtual world, for example web crawlers or the components within CyberDesk [Wood, 1997]. Between these we have more complex physical devices, such as the PDA, which both have a real-world existence and also serve as windows into virtual space (especially when combined with mobile communications).

To some extent context-awareness can be seen in hardware-configuration architectures such as Jini and Plug&Play. In both the emphasis is on self-discovery and automatic re-configuration of software to reflect the current hardware. The main difference between these and 'real' context-aware applications is one of time – the rate of reconfiguration required by, say, Plug&Play on a personal computer may only be a few times during the lifetime of the device. However, as inter-device protocols and standards such as BlueTooth become more prevalent hardware reconfiguration will become far more frequent and will be an important source of contextual information for higher-levels of the interface.

As we consider taxonomies and then models of space and location in sections 4 and 5, we explicitly consider both physical and virtual location and endeavour to construct theoretical and computational models which encompass both.

4. A design framework for mobile systems

The core of our framework for understanding the design of mobile systems is a series of taxonomies that consider the relation between different devices and the spaces they inhabit using location as a starting point for this consideration. Rather than seek to understand all senses of mobility for all potential forms of space we will particularly focus on the physical space of devices as a distinguishing feature. However, we will also draw on examples of the virtual where they are instructive to highlight the co-existence of these two forms of space and the issues of mobility that may exist in both. Although it is worth stressing that our understandings of this virtual space is still under development. We will also return to the issue of the real and the virtual when we consider the development of a model of space to support mobile systems.

As described in section 3.1, the core of our framework is an understanding of:

- location in space (of the device and other bodies)
- mobility through space (of the device and other bodies)
- the kinds of bodies populating the space (which the device may interact with)
- the awareness (of the device) of these other bodies

In this section we will develop taxonomies of location, mobility and population in turn, then finally a decomposition of types of device awareness.

4.1 A taxonomy of location

Mobility makes us think automatically about location, the way in which this sense of location can be understood in the system, and how changes in location can affect the system. Any simple mobile device will have a physical location in space. It is important to understand the nature of this location, and how the developers of interactive mobile applications may exploit this understanding. In this section we wish to consider what we might actually mean by location in space. This brief exploration is more than a mere issue of terminology as developing a understanding of what we actually mean by location represents a consideration of one of the core design concepts in the production of mobile systems.

Looking at the spatial dimension, there are some devices (for example GPS-based map systems) where the exact Cartesian position in 2D or 3D space is important in defining a sense of absolute physical location. For others a more topological idea of space is sufficient in understanding position and in these cases location is considered not in an absolute sense but in relation to other objects or sensors. For example the Lancaster GUIDE system is based on radio cells roughly corresponding to rooms and sections of Lancaster Castle and the CyberGuide [Long, 1996] system at Georgia Tech. shows visitors around the GVU laboratory by altering its behaviour depending on what item of equipment is closest.

In section 5 we will look at more formal models of space in greater detail, for now we will just use this two-way distinction between Cartesian space and topological space and consider location in these terms. As we discussed in section 3.2, it is important to consider location in both a physical and a virtual sense. If we consider ideas of virtual location, for example position within a hypertext, we see that we may have similar ideas of space within the electronic domain. This consideration of location provides us with the following simple taxonomy:

		Real	**Virtual**
space	Cartesian	GPS	VR
	Topological	room	hypertext

Figure 3. Location in different kinds of space

Note that these are not mutually exclusive categories: an item in a room also has a precise longitude and latitude, also a computational entity may have an existence in one or more virtual spaces as well as physical space. Indeed, possibly many of the most interesting

interaction possibilities occur when these different ideas of location are linked. For example, moving a display up and down in physical space could be used to change the display of hypertext help system for the maintenance of a piece of machinery. Similarly, in an aircraft cockpit, setting the destination city (topological destination) instructs the autopilot to take an appropriate course in Cartesian space/time. This interplay between the real and the virtual is central to the development of augmented reality spaces where the movement of devices within a space may manifest in effects that are both real and virtual. These spaces only work because the location of the device can be controlled in virtual and physical space and its effects provide alterations to either the physical or virtual space.

4.2 A taxonomy of mobility

Our core concern in the development of our design framework is the issue of mobility and its implications for how we understand human–computer interaction. In the previous section we considered how the issue of location can be unpacked to provide understanding in both a physical and a virtual sense and how the nature of the space affects our consideration of location. In this section we wish to focus on how to understand mobility and what potential design issues may emerge from a more detailed consideration of mobility.

Devices may be mobile for a number of reasons. They may be mobile because they are carried around by users (as with a PDA or a wearable computer), because they move themselves (robots), or because they are embedded within some other moving object (a car computer). Furthermore, a number of different devices may be spread within our environment so that they become pervasive, as in the case of an active room such as the ambient room suggested by Ishii [1997]. The issue of pervasiveness is itself a rather thorny one in that it is not clear what constitutes pervasiveness in terms of devices and how this relates to previous discussions surrounding ubiquitous devices. Ubiquitous computing has focused on the backgrounding of the device and the computer essentially "disappearing" into the environment. For us the issue of pervasive devices has less to do with the devices fading into the environment and more to do with an expectation that devices are normally available. Pervasive computing is intimately bound up with the inter-relationship between different devices and the expectation that these devices can work together to provide some form of shared functionality. An active room is active because it contains a number of devices which when they work in unison provide some function. Essentially, we are seeing a number of computing devices which in cooperation provide some functionality. Some of these devices may be mobile, but many are not. Consider, for example, the layout of radio-LAN base stations for the GUIDE tourist information system. These base stations have a fixed location in Lancaster, but are the source both of location and other information displayed on mobile devices. Neither the base stations nor the mobile devices can function by themselves, but together they allow the space to offer a pervasive computing facility.

We can disentangle the different levels of mobility into three dimensions that are used in Figure 4 to classify examples of mobile systems.

First, we can consider the *level of mobility* within the environment, this divides into three main categories:

- *fixed* – that is the device is not mobile at all! (e.g. a base station fixed in a particular place)

- *mobile* – may be moved by others (e.g. a PDA or wearable computer that is carried around)
- *autonomous* – may move under its own control (e.g. a robot)

The devices *relation to other devices* or its environment provides our second dimension and can also be divided into three different categories:

- *free* – the computational device is independent of other devices and its functionality is essentially self contained.
- *embedded* – the device is part of a larger device
- *pervasive* – the functionality provided by the device is essentially spread throughout the environment.

These separations do not consider the nature of the device and the sort of functions it may afford. The physical design of the device itself is an issue that needs to be considered carefully, especially in terms of existing traditions of aesthetic and practical design. The consideration of these features is beyond the scope of the framework and taxonomy we wish to present here, which focuses on the development of the device.

As a final part of our taxonomy we can reflect the cooperative nature of advanced mobile applications by considering the extent to which the device is bound to a particular individual or group. We have three classes for this too:

- *personal* – the device is primarily focused on supporting one person
- *group* – the device supports members of a group such as a family
- *public* – the device is available to a wide group

We do not suggest that these categories are absolute but rather provide them as sample equivalent cases of utility to designers. All the categories have grey cases, but perhaps this last dimension most of all. In particular we should really consider both the static and dynamic nature of how these categories are applied. For example, we could classify a computer laboratory as 'public', but of course, after logging in, each computer becomes personal. We will return to these dynamic aspects when we look at how devices can become aware of their users.

In fact, the 'group' category really covers two types of device. Some, like a liveboard [Abowd, 1998] actually support a group working together. Others, like an active refrigerator (which allows messages to be left, email browsing, etc.), may primarily support one person at a time but are available to all members of a family. In-car computer systems exhibit both sorts of 'groupness': they may perform functions for the benefit of the passengers of the car as well as the driver, and also the exact mix of people from within the family (or others) in the car may vary from trip to trip.

		Personal	Group	public
Free	Fixed	office PC	Liveboard	computer lab.
	Mobile	PDA	tour guides	
	Autonomous			factory robot
Embedded	Fixed		active fridge	ATM
	Mobile	wearable devices	car computer	shopping cart
	Autonomous		auto pilot	monorail
Pervasive	Fixed		active room	
	Mobile			Star Trek
	Autonomous	web agent	HAL	Web crawler

Figure 4. A taxonomy of different levels of mobility

Some of the examples in Figure 4 are clear, but some may need a little explanation. The 'Star Trek' reference is to the computer in Star Trek that responds to voice commands anywhere in the ship, but does not actually control the ship's movements. This pervasiveness of interaction is also evident in the work on ubiquitous environments developed by those at PARC [Wand, 1995]. Here the computational infrastructure is considered to be continually available. In a similar vein, we put HAL (the computer from 2001) in the group category as it has a small crew, but this is exactly one of the grey distinctions in constructing a taxonomy of this form. Our reference to 'shopping cart' refers to the development of smart supermarket trolleys that allow shoppers to scan the barcode of items as they are added to the trolley and keep track of your purchases to enable a fast checkout. Often these require the insertion of a shopper identification, in which case they become dynamically personalised.

Notice there are various blank cells in this taxonomy reflecting our use of it as a means of charting the design space for interactive mobile devices. Some of these blanks represent difficult cases where there may not be any sensible device. For example, a fixed–pervasive–personal device would have to be something like an active hermit's cell. In fact, the whole pervasive–personal category is problematic and the items 'web agent' and 'web crawler' in the final row may be better regarded as virtual devices of the free–autonomous class.

Other gaps represent potential research opportunities. For example, what would constitute a free–mobile–group device? This would be a portable computational device that supports either different individuals from a group, or a group working together – possibly an electronic map that can be passed around and marked.

As we suggested at the outset of our discussion of the design framework most of the examples are of physical devices. Virtual devices may also be classified in a similar way; for example, Word macros are embedded–mobile (or even autonomous in the case of macro viruses!) as are Java applets. The only virtual devices in Figure 4 are the items 'web agent' and 'web crawler' in the final row which, as we have said, may be regarded as virtual devices of the free–autonomous class. This ambiguity is because any virtual device or agent must be

stored and executed upon a physical computational device and the attributes of the physical device and virtual device may easily differ. For example, a PDA may contain a diary application. This is mobile by virtue of being stored within the PDA (a virtual device embedded within a physical device). However, if the PDA is used as a web browser it may execute a Java applet that is a form of virtual agent embedded within a web page (a virtual embedding in a mobile artefact). That is, we have an embedded–mobile–public virtual agent temporarily executing on a free–mobile–personal device! This dual presence in multiple contexts is both the problem and the power of virtual environments and will require significant further research to resolve.

4.3 A taxonomy of population

In addition to having a location in a space and exhibiting some degree of mobility, devices also need to be aware that they populate a space and need to reflect the coupling with the space they inhabit depicted in Figure 1. This awareness may include both the physical nature of the space (light, temperature, weather) and the electronic environment (network state, available memory, current operating system). A simple example of virtual devices understanding the space they inhabit are Javascript web pages that run different code depending on the browser they are running on.

Spaces are normally populated with a range of different devices. Within the physical and virtual spaces of a device there may be other computational devices, people (including the user(s) of the device) and passive objects such as furniture. These may be used to modify the behaviour of the device. For example, in CyberDesk 'ActOn' buttons are generated depending on what other applications are available and the types of input they can accept [Wood, 1997]

We can consider the issue of population in terms of the sorts of *bodies* that populate a space and the different spaces they populate. Figure 4 gives examples of items in the environment that may be relevant for a mobile or context-aware device and the bodies that may populate the space. In order to illustrate the development of this taxonomy we have taken a car computer and an active web page as two simple running examples.

	Physical (e.g. car computer)	Virtual (e.g. active web page)
People	current driver of car	visitor at web page
Devices	other cars	running applets
Objects	roadside fence	other pages on the site

Figure 5 Examples of bodies within the environment

4.4 Measurement and awareness

Each of the three taxonomies developed in the framework relies on devices having an awareness of the surrounding space and using this as a resource in interaction. The central role of awareness of the surrounding environment and how this awareness is conveyed to

others is an issue of some sensitivity in design. For example, in the case of active badges the issue of awareness of users and how this may be applied became embroiled within a discussion of privacy [Harper, 1992]. This may become even more problematic in the case of multiple devices that display an awareness of others. Consider the suggested “fun” interest badge devices offered by Philips in the development of its visions of the future design study [Philips, 1996] or the " Meme tags" developed at MIT [Borovoy, 1998]. These badges are programmed with a set of profiles for people and are intended to light up when you meet someone else with a compatible profile. The social acceptability of this form of device may well become a significant issue in determining their success and the general acceptance of devices of this form. For example, they have proven successful in conference settings [Borovoy, 1998].

In order to have an awareness of their environment, devices must be able to detect or measure significant attributes (for example, we have mentioned their location, environment, other devices, people and things). The discrete nature of computation means that both specification and implementation of computer systems tends to focus on events that occur at specific times. However, most of the contextual information above is of a different kind – *status* phenomena, that is they are things which constantly have a value that can be sampled. The translation of status phenomena into events is problematic and is often done 'accidentally' within systems, with the consequent probability of errors. Status–event analysis is a collection of techniques that lay equal weight to events and status. In particular, one strand of previous work in this area has looked in detail at the ways in which active agent can become aware of a status change [Dix and Abowd, 1996; Ramduny,1998]. In short, these reduce to finding out *directly* by its own sensors or *indirectly* via another agent (human or electronic). For example, a car with a built-in GPS sensor can detect its position directly and thus give directions to the driver, but a simple PDA may need to be told of the current location by its user in order to adjust time zones. Other computational agents may also be important sources of information about themselves (as in the case of onCue, where objects register themselves with the system) and about other parts of the environment (for example recommender systems, which say "others have visited here").

This leads to the two-way table (figure 6). The first axis is *how* a device finds out about contextual information: *directly* using own sensors measurement or *indirectly* told another device/user. The second axis is *what* is being discovered: the device's own attributes (location etc.), the attributes of another device, and, in the case when told indirectly, whether that device is telling of its own or a third party's attributes.

		how	
		direct – own sensors	indirect – told by others
what	own attributes (self)	GPS	PDA – location set
	other bodies	proximity sensors	object registration
	other (third party)		Recommender

Figure 6. Types of measurement and examples

Items in the environment (people, devices, objects) are particularly difficult: not only may they change their attributes (position, etc.), but also the configuration of items may change over time (e.g. people may enter or leave an active room). This leads to three levels of awareness. We'll look at these with the example of a car computer:

- presence – someone has sat down in the driver's seat, but all the car can tell is that the door has been opened then closed
- identity – the driver enters her personal pin number and the car can then adjust the see position for the driver
- attributes – the car detects from the steering behaviour that the driver is getting drowsy and sounds a short warning buzzer

Notice how, in this example, presence was not detected at all, identity was informed by the driver, but the sleepiness of the driver was detected directly. In other cases different combinations of detection or informing may be found. Security systems often have ultrasonic sensors to tell that someone is near (presence). Similarly, the car could be equipped with a pressure sensor in the driver's seat. Active badges, video-based face recognition or microphones matching footstep patterns can be used to tell a room who is there and hence play the occupant's favourite music and adjust the room temperature.

These examples are all about detecting people, but the same things occur in other settings. In the virtual world an agent may need to detect the same levels of awareness: presence – whether any other applications are running; identity – if so what they are (e.g. Netscape); and attributes – what web page is currently being viewed. Also, physical devices may detect one another, for example allowing several people with PDAs to move into 'meeting' mode. In fact, awareness models that do just this form of detection within the virtual world abound [Rodden, 1996].

Figure 7 summarises these various factors laying out awareness levels against the 'what' from figure 6. Differences between direct/indirect measurement are drawn out where relevant. Perhaps most interesting is the 'presence' row. There is no need for a device to measure its own presence – a computational equivalent of *cognito ergo sum*. Also, in the attributes, we have distinguished internal attributes (memory state of device) from external ones (sound coming from the speaker, position in space). Again this is because, a device is implicitly able to be aware of the former (although may not be in practice), whereas external attributes need some form of physical sensors. This state of affairs is reversed when looking at other bodies. Note that in this table the word 'announcement' means some sort of directed or broadcast communication between the other body and the device.

	self		**other bodies**	
	direct	indirect	direct	indirect
presence	implicit	n/a	proximity sensor	announcement
identity	Intel chip!	DHCP	Identifying attribute	announcement
internal attributes	implicit	n/a	only if explicitly represented in external attributes	announcement
external attributes	autosensors e.g. screen Pantone colour matcher	Windows message "did that screen setting work?"	sensors	autosensing + announcement

Figure 7 Taxonomy of device awareness of self and other bodies

In all the cases considered, detection and measurement may vary in accuracy: perhaps a box was put onto the car-seat pressure sensor, the driver lied about her identity, the ultrasonic sensor cannot tell whether there is one person or more. There will also typically be some delay, especially when indirect means are used, which is especially problematic if the attribute being measured changes rapidly. Thus actual detection is a trade-off between accuracy, timeliness and cost. Depending on the outcomes certain adaptations may be ill advised – a car wrongly identifies its driver and adjusts the seat thinking the driver is short, the real driver is quite tall and ends up squashed behind the steering wheel). The fidelity of awareness is very closely tied to the demands of the application and represents a genuine trade-off between the cost of measurement, the nature of the measurement and the importance of accuracy in the awareness information.

In developing the framework in this section we have explored the overall design space and suggested some ways in which we might characterise it. This characterisation forms the basis of the development of a model of space that supports mobile devices in maintaining an awareness of others reported in the following section. This model builds upon the taxonomies of location (figure 3) and bodies (figure 5). This focus is partly pragmatic and partly intrinsic: we cannot computationally model mobility until we model location, we cannot model awareness of other bodies in close locations until we have modelled bodies and their location. So, for the purposes of this paper, the taxonomies in figures 4, 6 and 7 will inform our discussion, but will not be explicitly represented.

5. Developing supporting models

The focus on the characterisation in the previous section has been a sense of location and mobility in space. In developing this characterisation we have concentrated on physical space while suggesting that a significant feature of the interactive nature of mobile systems is that they tie together different forms of virtual space without elaborating on the nature of these spaces. One reason for this is that while considerable agreement exists on the basic structure

and nature of physical space a similar general model of electronic spaces has yet to emerge.

To address this problem we have developed a general model of space that supports mobile devices in maintaining an awareness of others. Not surprisingly ideas of space and location are of critical importance in developing such a model. In order to adapt itself to its location, a mobile device needs to be able to ask:

i. Where am I?

ii. What else is nearby?

iii. How should I behave in the light of (i) and (ii)

And in the case of autonomous devices:

iv. How do I get to where I want to be?

We'll concentrate on the models of location needed to answer the first two questions as they final two questions are highly application specific.

Actual space and represented space

In practice, a device does not access the 'actual' location of itself or other objects directly, instead it accesses some computational representation of location held by itself or some sort of location service. Some form of transducer relates the 'actual' space and the representation of the space. This actual vs. representation is not just an issue for physical space, but also virtual space. An awareness mechanism on a web server may tell you about other current visitors to the site based on a site login/logout. However, some of the current 'visitors' may have simply omitted to logout before going to pages on another site. Thus the server's representation of the virtual location of those users is not their actual location in this virtual space.

To represent this separation between the actual space and representation of the space we need two kinds of model:

- *a semantic model* which can be used for both the actual space and the computational representation of the space
- *a computational* model which is part of the run-time architecture

The semantic model gives a common meaning to the actual and representational space and so allows us to discuss issues in the mapping between them including the fidelity of that mapping.

Although we need to deal with different kinds of space, if they are suitable for questions of type (i) and (ii), they must share some idea of location and some idea of *nearness*. There are several mathematical models of space that are informative as well as implicit models of space in various awareness models. In both types of model we will find explicit representations of *nearness*. We will briefly review these existing models and use these together with the taxonomies from section 4 to inform our construction of a semantic model, which in turn, will be instantiated in our computational model.

5.1 Existing models of space and awareness

We all feel we have some knowledge of ordinary physical space and those with a scientific background are used to encoding this in the x,y,z coordinates of Cartesian geometry. The Cartesian view of physical space allows a unique labelling of space and allows us to understand the relationships between locations in terms of their coordinates alone. Scientifically it has been of tremendous importance and practically it enables global navigation and the civil construction. In virtual reality it is this Cartesian 3D space that is emulated and in desktop interfaces Cartesian 2D space. One of the requirements we have is to have a measure of nearness and Cartesian geometry supplies this with the familiar Pythagorean (as the crow flies) distance:

$$dist^2 = x^2 + y^2 \quad \text{– 2D space}$$

$$dist^2 = x^2 + y^2 + z^2 \quad \text{– 3D space}$$

The awareness model of Benford et al [Benford, 1995] was designed to deal with proximity and attention in shared virtual environments. It is thus formulated within a strongly Cartesian spatial framework. Some of the concerns driving this model were pragmatic: "how can we know when an object is not the centre of a user's attention and so render it in less detail?", "how can we know to whom to transmit a particular user's audio so as not to drown everyone in a uniform babble?". The concepts of *aura*, *nimbus* and *focus* (and in later work *third party objects*) introduced in this model capture a relative notion of 'nearness': "what can I see/hear?". The fact that this is set within a Cartesian virtual reality environment means that there are already clear 'nearness' clues given by the scaling of objects with distance.

Despite its influence and conceptual power, Cartesian geometry is not as universal in the physical world as first appears. Cartesian coordinates are themselves built upon Euclidean geometry, which for almost 2 millennia was seen as self-evident. It was only comparatively recently (17C) that alternative regular geometries were discovered: spherical geometry (the surface of a sphere, where there is too little 'space' as one moves farther away) and hyperbolic geometry (where there is too much 'space' as one looks further away - cabbage leaf geometry!). Still more recently with general relativity it has become clear that large scale space is neither Euclidean not regular, but instead 'curves' as it is influenced by anything and everything that has mass or energy. At the quantum level things are still worse and it appears that space may become fractal.

In mathematics there are a number of fields of study aimed at understanding alternative kinds of space. Important historically was the study of the geometry of regular spherical and hyperbolic space, following in the same vein as traditional geometry with theorems about triangles, circles etc. and a whole study of spherical trigonometry. More interesting for virtual environments are various kinds of 'space' that are less regular and embody more abstract notions of 'nearness'. Two common abstract mathematical models of space that capture aspects of nearness are Metric Spaces and Topological Spaces.[1] Both of these

[1] Note that when we used the word 'topological' in section 4.1, it was in the weaker sense in which it is used in computing rather than the precise mathematical formulation of a 'Topological Space' (which will be capitalised to void confusion).

abstract mathematical spaces capture an idea of nearness. In the case of Metric Spaces this is a numerical measure of the distance between two points which satisfies the 'triangle inequality':

$$dist(a,c) + dist(c,b) \geq dist(a,b)$$

This effectively says that if you want to go from A to B, it is always as fast or faster to go directly rather than to stop of at some other place C on the way – a reasonable minimal property of distance.

In the case of Topological Spaces, the idea of nearness is captured by ever-decreasing 'neighbourhoods' which contain a point and all sufficiently close neighbours. In both these kinds of spaces, the main interest in mathematics is in the notion of series of points which get ever closer without reaching a given point (convergent sequences), they are treated like 'rubber sheets' which can be stretched as much as you like so long as they aren't torn (continuous mappings). Hence, even in the case of Metric Spaces which have a numeric measure of distance, the important factor for their mathematics is not the absolute measure of nearness, but the use of the numbers to see whether things are getting closer.

More abstract notions of space can be found in Rodden's formalisation [Rodden, 1996] of the Benford et al awareness model. The spatial model underlying Benford et al's work was clearly Euclidean, but largely implicit. Rodden's work looked at awareness over a graph structure as is found in the web and many other computational domains. Nearness in such a space can be measured by number of arcs traversed or similar weighted measures both of which yield Metric Spaces. However, the critical properties of nearness in this work do not depend on these particular properties of the underlying graph. This suggests that we need models of space which may be stronger in that we would like some absolute sense of nearness and weaker in that we don't need the complex mechanisms needed to discuss convergent sequences etc.

A final form of mathematical 'space' which is relevant is the Differential Manifold. This is used to model curved space-time in General Relativity. This is not directly relevant as a model of the kinds of location found in virtual space or much-slower-than-light-speed physical space. However, the ways in which relativity has challenged our understanding of 'space' in the physical world has a lot to teach us about the challenges of 'virtual' space. One particular point is the way general relativity models space using mathematical structures called Differential Manifolds. Because space curves and may have 'singularities' (such as black holes) and even distant linked points (wormholes), it is impossible to use a single co-ordinate system to refer to all points. Instead, the models consist of a number of patches, each of which has 'ordinary' Cartesian co-ordinates. Where the patches overlap there is a gentle transition between the co-ordinate systems (in mathematical terms they are related by a smooth function). Virtual spaces, such as the web, may similarly have no global map or model, but if we can establish patches with well defined structure and clear transitions between them then there is some hope for lost users.

Not only is space in general relativity not flat, but its shape and 'size' change in time. We have all heard of the expanding universe. This doesn't mean simply that the stars are flying apart through space, but instead that the *space itself* between the galaxies is stretching. This at first sounds as if it is only of interest to cosmologists. However, it is also precisely the

experience of those using wireless communications when their connection is broken. Before the break in communication they have established a sense of 'nearness' in virtual space with other people and things on the network. Then when the connection breaks this virtual geometry suddenly changes – things that were near suddenly become far away. It is precisely the difference between this and 'normal' space that makes such disconnections so disturbing, especially if a collaborative system engenders any sense of immersion.

The feature of space in General Relativity that is perhaps most well known (although not necessarily understood) is that time and space are dealt with on an equal and interlocked basis: the time–space continuum. This blending of time and space can also be found in more mundane areas of virtual environments and interface design.

The Aether model of Sandor et al.[Sandor, 1997] adopts a graphical network as its underlying space, very like Rodden's model. However, whereas both Benford et al and Rodden have declarative definitions of awareness, the Aether model adopts a more process-oriented mechanism whereby the influence of an object (*aura*, *nimbus* and *focus*) percolates through the network, getting weaker as it passes from node to node. The choice of this mechanism was largely driven by implementation considerations of producing an 'awareness engine', but is, of course, very like the physical transmission of sound and light. The Aether model has an implicit measure of nearness given by the rate at which network links and node attenuate influence, but also the Aether model explicitly introduces time as part of its awareness model. Whether in physical space or virtual, as soon as one takes into account transmission delays, space and time become inseparably interlinked.

This interlinking of time and space also becomes important as we consider different sensory experiences [Dix, 1996]. Different senses give us different 'cuts' through time space. For objects within sight, we can consider the speed of light as practically instantaneous. Hence a quick glance around tells you about an area of space at a particular instant in time. If you want to know where something was a few seconds ago, you need to have looked then and remember. Imagine, however, that you are a dog or mole and are working using a sense of smell. As you sniff at a particular location you get some idea of the various creatures that have passed and even recent weather conditions at that point. That is smelling tells you about recent time at a single point of space. If you want to know what happened at other locations you need to have smelled there and remember. Finally consider a creature that uses sonar such as a whale or bat. Because sound takes time to travel through water or air, he echoes heard at a single moment correspond to close things recently, but further things longer ago. Figure 8 shows how each of these give us a particular cut of space-time.

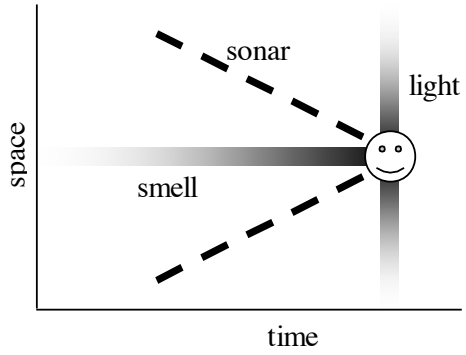

Figure 8. Different cuts through space-time

In virtual space, network delays mean that we have sonar-like mixing of time and space. But also computer systems embody a memory of interaction – traces of past commands, windows opened for a previous purpose and never closed, copies of possibly out-of-date information – more like the world of smell. To add to the confusion, these different cuts through time-space are typically all presented visually!

In order to be able to talk about time-space interactions precisely, we need an semantic model of space (virtual ad physical). Before we consider such a model it is worth highlighting that we will not be able to investigate all the aspects of time-space in this paper and that the model presented in the following section is only one part of a much richer picture.

5.2 Developing a semantic model of space

As discussed, existing awareness models are focused primarily on virtual space taking lessons from physical phenomena and are based on different underlying models of space. In order to support this range of approaches to awareness we need an abstract model of space that includes both Euclidean space and network space. We do not need the full richness and complexity of the mathematical spaces, but we do need an explicit formulation as we need to be able to talk about several simultaneous spaces and their relationships.

Kinds of space

The fundamental concepts we require are *location* and *nearness*. So we define a 'Space-Kind' to include precisely these two plus some functions relating them:

Space-Kind = Location – set of elements representing 'locations'
Nearness – partially ordered set of 'nearness' values
dist: Location × Location → Nearness
…

The Location set depends on the precise kind of space. In 2D Cartesian space it would be the set of all (x,y) coordinate pairs, on the web it would be the set of all URLs, in a building it would include locations such as 'floor 3', 'room A 312', or 'north-east stairwell'.

The Nearness set will normally contain a minimal element 'HERE' representing 'at the same place as'. In the case of Cartesian space it is simply positive real numbers and 'dist' would be

the normal as-the-crow-flies distance. In other spaces Nearness is a less precise concept and has values such as, 'on the same web page', 'at the same site as', 'in the same room as', 'in the same building as'.

The Nearness measure need not be a total order. For example, in a geographical information system we may be able to locate roads within towns and so 'in the same road as' is obviously closer than 'in the same town as'. However, if we look in the countryside it may simply be able to tell us 'on the same mountain as'. Is this a closer measure than 'in the same town as'? Although this Nearness set is intended to give some idea of absolute distance, we need to be careful. A very clear lesson from the mathematical studies of metric and topological spaces is that non-local measures of distance need to be treated with extreme caution. In our context we want to be able to conclude that if A is a device and X and Y are objects in a space such that:

$$dist(A,X) < dist(A,Y)$$

It is fair to make A's behaviour more dependent on X's presence than that of Y. However, if A and B are devices such that

$$dist(A,X) < dist(B,Y)$$

We should be extremely cautious about making any strong statements about the comparative strength of the relationships A–X and B–Y.

This does not mean we never use absolute judgements of distance. We have to be able to say things like:

> If object X is in the same room as device A, then A shows a representation of X in its screen.

However, when designing such rules we have to be aware that 'in the same room as' could mean a broom cupboard or an auditorium.

Some kinds of location have a natural idea of containment. In an office complex 'room A 315' may be on 'floor 3' of 'building A'. Similarly, the hierarchy of a web site leads to a natural hierarchy of locations. In Cartesian spaces locations are mutually exclusive, but one can have regions of space, for example 'all points within 3 miles of Great St Mary's Church Cambridge'. In order to capture both these uniformly we allow a space to have a set of regions which may either be a subset of locations (in the case of a hierarchy), or represent well formed sets of points (in the case of a Cartesian space) or some other domain specific concept:

Space-Kind = …
Region – sensible areas
contains: Region × Location → Boolean

Spaces and bodies

As we have already noted, even in the physical world we have several simultaneous ideas of space: longitude and latitude, town-street etc. In the virtual world this multiplies further. So our model of the world has a number of spaces each of a particular kind and with other domain specific attributes:

World = Spaces – set of elements representing 'locations'
kind: Spaces → Space-Kind
attr: Spaces → Attributes
…

People, devices and passive objects also inhabit the world as we have discussed previously. We call these collectively 'Bodies'. These again have various domain specific attributes, but the crucial question is the location of a specific body. This is not absolute, but defined relative to a secific space (e.g. GPS coordinates):

World = …
Bodies – (Bodies = People + Devices + Objects)
attr: Bodies → Attributes
loc: Bodies × Spaces → Location

The 'loc' function is partial as there may be 'spaces' for which a particular body have no clear relationship: for example, there is no sensible answer to the question "what URL (web location) is my tea cup at?".

5.3 A computational model

Our semantic model of space is based on the concepts of *Space* & *Bodies* and a representation of location. We have developed a corresponding computational model to allow mobile applications to share a common awareness of a space and the bodies that inhabit that space. The general approach is to use an object-oriented model with a small number of simple objects that can be made shared across a distributed information space. This allows the state of defined objects to be accessed by a number of different devices.

The core of the model depends upon the definition of a virtual model of space in which the bodies relevant to the system are located and the ability to reason about the location of these in terms of a developed virtual space and the physical space of the real world. The central elements in the computational model are a *world object* and a *body object*. Each of these objects are intended to provide the root of two distinct specialisation trees to represent the different forms of space and the bodies that exist within the space. These two object hierarchies essentially instantiate the different kinds of space and bodies suggested in the semantic model. This development of a number of models of space mirrors the taxonomy we developed in the design framework and provides us way of reasoning about the location of devices in mobile systems in terms of both a real and virtual locations.

All the objects are realised on top of a distributed platform that allows the state of Java objects to be shared between different applications. The classes introduced below are therefore subclasses of *SharedEntity*, which is the root of all objects shared across the distributed platform.

The Space Object

The space object focuses on the ability of a space to act as a container of objects and on the way in which space can structure the world in terms of containment. The core space object has only two significant attributes a set of bodies that it contains and a set of locations for these objects in the space. Location depends on a location object that can represent different

senses of location.

The space object is provided to developers as a Java class that can be extended and specialised in order to represent different forms of space. Updates to the space object are propagated to all those that have registered an interest in the space. The key elements of the space object are reflected in Java as:

```
Class Space extends SharedEntity {
        Kind kind;                      // The kind of space (e.g Cartesian or Topology..)
        Vector bodies;
        Vector locations;               // corresponding locations of bodies
        … }
```

The core *Space* class includes methods for adding and removing bodies to a space, finding the location of a body within the space and moving bodies in the space. Specialisations drawn from this core object exploit the semantics of the space to provide more sophisticated views of proximity and distance.

The Location Object

A location class handles the location of bodies in space and is closely associated with the space object: each location object has an attribute to determine the kind of space it refers to and each body in the space has an associated instance of a location object. Each location object basically represents the more general structure of the space within which bodies are placed. The attributes of the core Location class are:

```
Class Location extends SharedEntity {
        Kind kind;                      // The kind of space (e.g Cartesian or topological)
        Vector connected_entities;      // The entities connected to this one
        Position position;              // The position in the space.
        … }
```

This basic location class has attributes that allow two different kinds of space to be represented: Cartesian spaces that define a location in terms of a position with reference to a fixed origin and topological spaces that consider the linkage between different spaces. This is also reflected in the fact that the space provides two distinct methods *position* that returns a 3D position for a given object and *connected* which returns a list of spaces that a given space is connected to. The *connected* attribute allows us to represent a range of graph-like spaces. Although there are other forms of non-Cartesian space these graph-like spaces include those most commonly found in information systems and the base class can easily be subclassed for other kinds of space. The *kind* attribute of the location should of course agree with the kind of space it is being used in and the methods provided by the core classes maintain this consistency.

The Body Class

The definition of the location class allows us to represent and reason about the location and position of bodies within any space. The core of our design framework was the need to consider bodies as having both real and virtual locations and to manage interaction in terms of the correspondence between these. This means that our computational model needs to

allow an interaction with bodies in terms of their position in multiple spaces. The link between the bodies representing the overall system and the spaces in which they reside is reflected in the definition of the class that provides the root of the bodies hierarchy making up the overall system. This is achieved in the computational model by having a Body class definition that allows state information about the spaces bodies are in to be externalised. The core attributes of the root Body class is a list of the spaces the body exists in. (Recall that a body may simultaneously exist in several physical and virtual spaces.) This is represented in the Java class as a simple vector:

```
Class Body extends SharedEntity {
        Vector Spaces;                  //  The spaces a body exists in
        … }
```

As in the case of the definition of the Space class the Body class inherits from SharedEntity. This means that the state information can be shared and made available across the distributed platform. Each shared entity has a unique name and a set of optional tags that can be used to find objects of particular interest. This arrangement allows the different components making up a mobile system to be aware of the location of other entities in the various spaces in which they reside.

Using the model

The computational model has been realised over a distributed infrastructure called Limbo (discussed in section 6.2). This platform allows search facilities over the distributed shared object store, thus allowing any application to find out:

- What spaces exist and what bodies are in those spaces
- What other entities are in locations close to a body
- What other spaces is this body in and what is it location in these spaces

This information can be used to infer distinct contextual cues about the nature of space. As an example consider a simple illustration of the use of the platform drawn from our experiences of the GUIDE project. A portable notebook has facilities that allow it to know which cell it is in a cell-based radio infrastructure. This is a topological space with a close correspondence to the physical arrangement of the devices in the real world.

This complete space can be represented as an instance of (a subclass of) Space called "physical radio" which has a Kind attribute set to "Topological". Each location in the space is named. Location names are based on the name of the base station supporting the cell. All the notebooks within the space are associated with a location and we can ask the platform for the bodies in the space and their location. The physical movement of notebooks is reflected as changes to the associated location in the "physical radio" space.

Each notebook's body definition also records the other spaces in which it is present and this can be exploited or even coupled with the information about its location in the physical space. For example, each of the notebooks in the GUIDE project shows a web page based on the radio cell that it is physically located in. This is achieved by putting the notebook in a virtual information space we shall call "guide space". This space is actually a set of connected web pages. The guide browser shows the appropriate page by finding its location within the virtual information space and updating this location as the location of the notebook in the

"physical radio" space changes.

The computational model allows us to represent a range of different models of space and location central to the contextual interaction underpinning mobile systems. This model can then be accessed by a range of mobile devices and shared between them. This shared computational model provides a higher level representation, which allows the rapid development and alteration of interactive applications essential to most prototyping approaches.

However, note that this computational model needed to be realised over an underlying infrastructure and system architecture (in our case the extended Limbo platform [Palfreyman 1999]). In the next section, we will discuss the various issues involved in designing and selecting such architectures and discuss how the Limbo platform meets these requirements.

6. From requirements to architecture

The taxonomies we have developed in this paper has highlighted a wide range of application niches and suggests many exciting design possibilities for specific applications exploiting the contextual nature of mobile devices. Although we are investigating some of these in a number of projects the primary aim of our current 'infrastructure' project is to examine the generic requirements to emerge from taxonomies of this form. These requirements can then be exploited to develop the underlying toolkits, architecture and infrastructure needed for temporally well-designed, context-aware, collaborative mobile systems.

Mobile systems extend our considerations of interaction beyond the user interface to consider interaction in terms of the entire environment (human, physical and computational). As our framework has highlighted, in mobile systems the relevant semantics includes issues of location in physical and virtual space; proximity of other devices and people; and capabilities of devices and communication infrastructure. The necessary information and functionality to exploit this context is typically widely distributed within the computational environment – spread over different devices, spread over different physical locations, and spread between different layers in the system. The individual application developer will simply not have the relevant information and functionality available unless the infrastructure is designed taking into account human interface requirements. Thus, in mobile systems more than other areas of HCI, the design of infrastructure is a central and essential concern.

6.1 Requirements

Unfortunately, research has repeatedly demonstrated the shortcomings of existing infrastructure components for supporting adaptive mobile applications [Davies, 1994], [Joesph, 1995]. In more detail, existing components have two critical shortcomings. Firstly, they are often highly network specific and fail to provide adequate performance over a range of network infrastructures (e.g. TCP has been shown to perform poorly over wireless networks [Caceres, 1994]). Secondly, existing components often lack suitable APIs for passing status information to higher levels. As a consequence of these shortcomings new systems are increasingly being developed using bespoke communications protocols and user interfaces. For example, the GUIDE system described in [Davies 1998] uses a broadcast-style protocol. This is appropriate in a location-based information system where it is likely that

pages of information needed by one device will be useful to all. It also uses the presence of base stations as an indicator of location, a technique shared with several current location-aware systems.

As these devices become more widespread the need increases for generic application architectures for at least specific subclasses of the mobile domain. There is clear commercial pressure for this, in particular, Windows-CE is being promoted for use in embedded systems. However, if these are simply developed by modifying architectures and toolkits originally designed for fixed environments there is a danger that some of the rich interaction possibilities afforded by mobile devices may be lost.

There are some examples of generic frameworks on which we can build. In Georgia Tech., location aware guides are being constructed using the CyberDesk/Cameo architecture [Wood *et al.*, 1997]. Cameo is a software architecture based on the theoretical framework of status–event analysis (as discussed in section 4.4). Because the phenomena we are trying to model are largely status, there is a great advantage of the underlying architecture also reflects this

Another major architectural issue for context-aware applications is the way in which contextual issues cut across the whole system design. This is reminiscent of other aspects of user-interface where the structures apparent at the user interface often do not match those necessary for efficient implementation and sound software engineering [Dix and Harrison, 1989]. In UI design this has led to a conflict between architectures which decompose in terms of user interface layers, such as the Seeheim and ARCH-Slinkey models [Gram and Cockton, 1996] and more functionally decomposed object-oriented models. In fact the object and agent-based architectures themselves usually include a layered decomposition at the object level as in the MVC (Model–View–Controller) model [Lewis, 1995] and in the PAC (Presentation–Abstraction–Control) model [Coutaz, 1987]. Although the display and input hardware may be encapsulated in a single object or group of objects, its effects are felt in the architectural design of virtually every user-interface component. In a similar fashion the hardware that supplies contextual information may well be encapsulated within context-objects, but their effect will permeate the system. This requires a similar orthogonal matrix structure similar to that found in models such as PAC or MVC. We expect context-awareness mechanisms to emerge as structures cutting across application layers and interface components.

Reviewing our discussion, an architecture for supporting mobile, context-aware applications must be:

- *distributed* – as this is the nature of the devices over which it operates
- capable of representing *location*
- able to effectively deal with both *status and event phenomena*
- be *othogonal* to other interface components

6.2 Extending Limbo to provide a supporting platform

Our computational model has been instantiated over a distributed platform that allows a number of devices to make state information accessible to each other and thus allows the

creation of a community of devices. This framework builds directly on the authors' previous work on Limbo [Friday, 1999] and the development of a shared interaction platform [Palfreyman, 1999]. The platform exploits a distributed tuple space to share state information between geographically remote clients allowing them to function as a single collaborating system. The developed infrastructure is constructed using a combination of C++ and Java and has four significant components:

- A distributed tuple space (Limbo) that allows tuples of data values to be shared between different devices and accessed across a range of communication facilities. Our particular tuple space is a mobile variant of the established Linda model [Gelernter, 1985].
- A notification services that informs applications about changes in the tuple space.
- An infrastructure that allows structured information to be mapped onto shared tuples.
- The set of specific Java context objects (as described in section 5.3) that allow communal access to the shared information about bodies, spaces and locations.

The general architectural arrangement is shown in figure 9. The general platform interface is provided through a set of Java objects. The implementation of the distributed tuple space allows the rapid replication of these objects and for changes in state to be propogated.

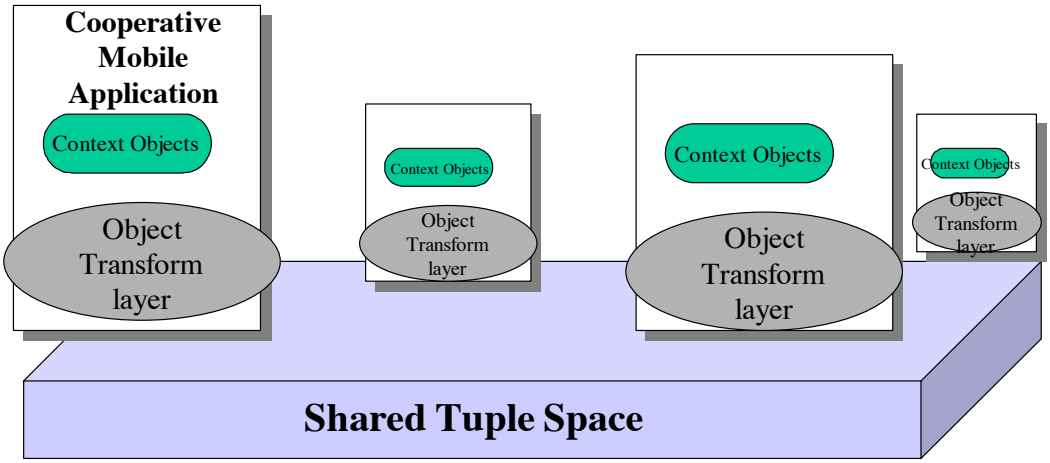

Figure 9. The platform Architecture

We will examine this architecture using the requirements identified in section 6.1.

Distribution

Limbo is one of a number of distributed platforms. Indeed, our use of Limbo has similarities to the recent emergence of JavaSpaces [Sun, 1998] which provides a tuple based infrastructure for Java programs. However, there are significant differences in the implementation resulting from the requirements that have driven their respective designs. Our platform is specifically designed for rapid dissemination of events and context information and this is reflected in the use of multicast within its underlying implementation. Limbo is

thus not only distributed, but optimised for the kinds of transactions we envisage for context-aware applications.

Location

The underlying Limbo architecture supports the sharing of any kind of objects. This, combined with our computational model of space, allows the representation and sharing between devices of:

- different elements of space,
- bodies and their locations,
- other domain specific objects

Together these facilities allow applications to provide interaction possibilities sensitive to context information pooled from a number of distributed sources.

Status and event phenomena

Limbo, the distributed Linda tuple space, upon which we have built our computational model of space and location, is already a status-orientated representation. The raw Linda space is a passive status requiring polling to discover changes. However, the additional 'Shared Universe' layer adds explicit event notification facilities allowing applications to react to status-change events and to use the platform for general event notification. Because of the ability of the platform to manage both status and event phenomena, we expect it to support other forms of context awareness as well as the location services explored in detail here.

Orthogonality

It is in recognition of the pervasive nature of contextual dependency within the interface that we have developed our shared space model as an underlying service rather than a widget or component. The distributed nature of Limbo means that the shared space model allows appropriate definitions of space to be shared between devices and for these devices to exploit this contextual information represented by this shared context. The orthogonality of the model to other infrastructure components means that it is capable of capturing location information from both low-level sources (such as a GPS data interface) or higher-level sources (such as a web browsers current page).

6.3 From theory to practice

In the time between when this paper was first written and this final copy was produced, one of the authors has been involved in the design and deployment of a new software product onCue (aQtive 1999). onCue is a rather unusual product, rather like an intelligent toolbar. It watches the user's current work context, in particular the contents of the clipboard, and suggests appropriate Internet services and desktop applications. For example, if the user copies a table onCue would change to include icons to suggest adding up the numbers, putting the table into Microsoft Excel, or visualising the table using an interactive histogram. If the user selects one of these, say the histogram, onCue automates the insertion of the data into the relevant application (figure 10).

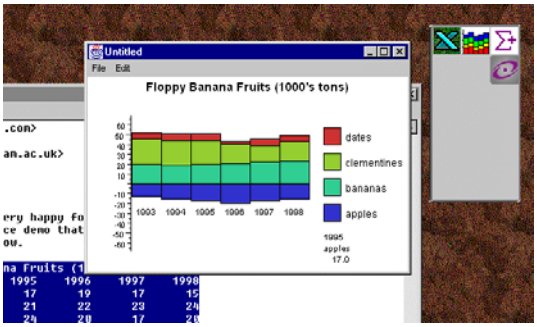

Figure 10. onCue suggestions for a table

onCue is clearly a highly context-aware application, but, in its current release, is neither location-aware nor mobile. This difference in design context means that onCue does not draw directly on the computational model and infrastructure that we have described. However, it has been heavily influenced by the wider conceptual issues described in this paper.

Following the principles in section 6.1, an appropriate architectural infrastructure has been key to the development of onCue, which is constructed over a platform called aQtiveSpace. Like the Limbo-based infrastructure described in section 6.2, aQtiveSpace offers a shared object space allowing components (called Qbits in aQtiveSpace) to be aware of one another and their capabilities. Also, status–event analysis has been one of the driving influences behind aQtiveSpace so it can easily deal with the many status-phenomena required for context-aware applications such as onCue. Finally, the open software architecture means that context-aware services are offered in a way that is orthogonal to the individual Qbits and furthermore includes non-onCue-aware applications within a coherent inter-application framework.

Although the different design contexts yield are reflected in the detailed implementation of aQtiveSpace and our Limbo-based infrastructure, the same fundamental theoretical analysis is embodied in both. This is clearly demonstrates the value and need for such conceptual frameworks and the importance of a general reflection on these issues rather than a more piece meal approach to the development of these fundamental issues.

7. Conclusion

This paper has consider the emergence and development of a new class of advanced cooperative application and the different forms of interaction that may need to be supported. The maturing of technology to allow the emergence of multi-user distributed applications that exploit mobile applications means that we can no longer focus the issues of interaction on the nature of the device. Rather we must explicitly consider impact of the context in informing the design of different interaction techniques.

In this paper we have focused on understanding the design space to emerge for this new class of application and the importance of location in mobile systems. The paper has presented a characterisation of this design space and for a particular portion of the design space described more detailed models and supporting platforms that reflect the general approach to

understanding the design space.

The general approach to developing our characterisation of the design space has been to focus on the central role of location in mobile systems. This focus represent only one of the potential contexts of relevance to mobile systems consider in section 2. The importance of space and location described in sections 3 and 4 respectively represent a fairly unique aspect of mobile systems and underpin the development of the more detailed models described in section 5 that are instantiated in the platforms described in section 6.

The use of location within this paper represents only one approach to understanding and managing context for these systems but as we said in section 2 the issues of context are much broader than location. In addition to the location of the device the overall context needs to be considered in terms of the devices relationship with the technical infrastructure, the application domain, the socio-technical system in which it is situated, and the physical nature of the device. The interaction style supported by mobile applications is as dependant on this context as the properties of the device itself. As a result, it is essential that work on the nature of these devices is complemented by a broader consideration of the nature of interaction. Our consideration of location and the development of the taxonomies, models and supporting platforms represents one step in this broader consideration.

References

Aliaga, D. G. (1997). Virtual objects in the real world. *Communications of the ACM,* **40**(3): 49-54.

aQtive limited (1999). onCue. http://www.aqtive.com/community/research/

Bass L., Kasabach C., Martin R, Siewiorek D, Smailagic A. and J Stivoric (1997); The design of a wearable computer ; proceedings CHI'97 Human factors in computing systems , 1997, Pages 139- 146

BCS HCI (1997). British HCI Group Workshop on *Time and the Web*. Staffordshire University, June 1997.

Benford, S., Bowers, J., Fahlen, L., Mariani, J, Rodden. T, Supporting Cooperative Work in Virtual Environments . *The Computer Journal*, 1995. 38(1).

Borovoy, R., Martin, F., Vemuri, S., Resnick, M., Silverman, B. and C Hancock (1998) Meme tags and community mirrors: moving from conferences to collaboration, Proceedings of the ACM 1998 conference on Computer supported cooperative work, pages 159-168

Cáceres, R., and L. Iftode. "The Effects Of Mobility on Reliable Transport Protocols." *Proc. 14th International Conference on Distributed Computer Systems (ICDCS),* Poznan, Poland, Pages 12-20. 22-24 June 1994.

Coutaz, J. (1987). PAC, an object oriented model for dialogue design. *Human\(enComputer Interaction \(en INTERACT'87*, Eds. H.-J. Bullinger and B. Shackel. Elsevier (North-Holland). pp. 431-436.

Davies, N., G. Blair, K. Cheverst, and A. Friday. "Supporting Adaptive Services in a Heterogeneous Mobile Environment." *Proc. Workshop on Mobile Computing Systems and Applications (MCSA),* Santa Cruz, CA, U.S., Editor: Luis-Felipe Cabrera and Mahadev Satyanarayanan, IEEE Computer Society Press, Pages 153-157. December 1994.

Davies, N., K. Mitchell, K. Cheverst, and G.S. Blair. "Developing a Context Sensitive Tourist Guide", *Technical Report* Computing Department, Lancaster University. March 1998.

Dix, A. and G. Abowd (1996). Modelling status and event behaviour of interactive systems. *Software Engineering Journal,* **11**(6): 334–346.

Dix, A. J. (1992). Pace and interaction. *Proceedings of HCI'92: People and Computers VII*, Cambridge University Press. pp. 193-207.

Dix, A. J. (1995). Cooperation without (reliable) Communication: Interfaces for Mobile Applications. *Distributed Systems Engineering,* **2**(3): 171–181.

Dix, A. J. (1996). Closing the Loop: modelling action, perception and information. *AVI'96 - Advanced Visual Interfaces*, Eds. T. Catarci, M. F. Costabile, S. Levialdi and G. Santucci. Gubbio, Italy, ACM Press. pp. 20-28.

Dix, A. J. and M. D. Harrison (1989). Interactive systems design and formal development are incompatible? *The Theory and Practice of Refinement*, Ed. J. McDermid. Butterworth Scientific. pp. 12-26.

Fickas, S., G. Kortuem, and Z. Segall. "Software Issues in Wearable Computing." *Proc. CHI Workshop on Research Issues in Wearable Computers,* Atlanta, GA, U.S.,

Fitzpatrick, G., et al, Physical Spaces, Virtual Places and Social Worlds: A study of work in the virtual, *Proc. CSCW'96*, ACM Press

Gaver W., The Affordances of Media Spaces for Collaboration, *Proc. CSCW'92*, 1992, ACM Press.

Gelernter, D.(1985) Generative Communication in Linda, ACM Trans. on Programming Languages and Systems, 5(1), pp 77-92.

Gram, C. and G. Cockton, Eds. (1996). *Design Principles for Interactive Software*. UK, Chapman and Hall.

Greenberg S.,Marwood D., 'Real Time Groupware as a Distributed Dystem; Concurreny Control and its effect on the Interface' Proceedings of CSCW'94, North Carolina, Oct 22-26, 1994, ACM Press.

Harper, R.H (1992). Looking at ourselves: an examination of the social organisation of two research laboratories. Proceedings of CSCW'92, Toronto, Canada, pp 330-337, ACM Press.

Henderson, A.J., and Card, S.A., Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention, *ACM Transactions on Graphics*, Vol. 5, No. 3, July 1985.

Howard, S. and J. Fabre, Eds. (1998). Temporal Aspects of Usability: The relevance of time

to the development and use of human-computer systems – Special issue of *Interacting with Computers* (to appear).

Hughes J., Rodden T., King V., Anderson K. 'The role of ethnography in interactive systems design', ACM Interactions, ACM Press, Vol II, no. 2, 56-65, 1995.

Ishii, H., and Ulllmer, B. (1997). Tangible Bits: Towards Seamless Interfaces between People, Bits, and Atoms. Proceedings of CHI '97, ACM Press.

Johnson, C. and P. Gray (1996). Workshop Report: Temporal Aspects of Usability (Glasgow, June 1995). *SIGCHI Bulletin,* **28**(2).

Johnson, C. W. (1997). The impact of time and place on the operation of mobile computing devices. *Proceedings of HCI'97: People and Computers XII*, Bristol, UK, pp. 175–190.

Joseph, A., A. deLespinasse, J. Tauber, D. Gifford, and M.F. Kaashoek. "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.

Lewis (1995). *The Art and Science of Smalltalk.* Prentice Hall.

Long, S., R. Kooper, G.D. Abowd, and C.G. Atkeson. "Rapid Prototyping of Mobile Context-Aware Applications: The Cyberguide Case Study." *Proc. 2nd ACM International Conference on Mobile Computing (MOBICOM'96),* Rye, New York, U.S., ACM Press,

Patterson, J.F et al., Notification Servers for Synchronous Groupware*, Proc. CSCW'96*, ACM Press.

Ramduny, D. and A. Dix (1997). Why, What, Where, When: Architectures for Co-operative work on the WWW. *Proceedings of HCI'97*, Bristol, UK, Springer. pp. 283–301.

Ramduny, D., A. Dix and T. Rodden (1998). Getting to Know: the design space for notification servers. *submitted to CSCW'98*,

Rodden, T., (1996) Populating the application: a model of awareness for cooperative applications, in: CSCW '96. Proceedings of the ACM 1996 conference on on Computer supported cooperative work, pp 87-96.

Root, R.W., Design of a Multi-Media Vehicle for Social Browsing, *Proc. CSCW'88*, Portland, Oregon, Spetember 26-28 1988, pp25-38

Roseman, M, Greenberg, S, TeamRooms: Network Places for Collaboration, *Proc. CSCW'96*, ACM Press

Sandor, O., Bogdan, C., Bowers, J, (1997), Aether: An Awareness Engine for CSCW, in J.Hughes et al (eds.) Proceedings of ECSCW'97: the Fifth European Conference on Computer Supported Cooperative Work, pp 221–236, Kluwer Academic Press.

Sun (1998), Java Distributed Computing White Papers, available at http://java.sun.com/products/javaspaces/whitepapers/index.html

Want R., Schilit, B. N., Adams, N. I., Gold, R., Petersen, K., Goldberg, D., Ellis, J. R. and M. Weiser (1995) An Overview of the ParcTab Ubiquitous Computing Experiment. IEEE Personal Communications, December 1995, pp 28-43.

Weiser, M. (1991). The computer of the 21st century. *Scientific American,* **265**(3): 66-75.

Weiser, M. (1993). Some computer science issues in ubiquitous computing. *Communications of the ACM,* **36**(7): 75-84.

Wood, A., A. K. Dey and G. D. Abowd (1997). CyberDesk: Automated Integration of Desktop and Network Services. *Proceedings of the 1997 conference on Human Factors in Computing Systems, CHI '97*, pp. 552–553.

Benford, S. Greenhalgh, C., Reynard, G., Brown, C, Koleva, B (1998). Understanding and constructing shared spaces with mixed-reality boundaries. *ACM Trans. Computer.-Human. Interaction. Vol. 5, No 3*, pp 185 - 223. ACM Press.

Palfreyman, K., Trevor, J., Rodden T. (1999). PSI: A Platform for Shared Interaction. Proceedings of ECSCW99, 12-16th September 1999, Copenhagen, Denmark, pp 351-371 , Kluwer Academic Press.