Res. Lett. Inf. Math. Sci., 2006, Vol 9, pp 13-87
Available online at http://iims.massey.ac.nz/research/letters/

Quality assessment technique for ubiquitous software and middleware

H. RYU[1], G.Y. HONG[1] & H. JAMES[1]

[1] *Institute of Information & Mathematical Sciences*
Massey University at Albany, Auckland, New Zealand [†]

The new paradigm of computing or information systems is ubiquitous computing systems. The technology-oriented issues of ubiquitous computing systems have made researchers pay much attention to the feasibility study of the technologies rather than building quality assurance indices or guidelines. In this context, measuring quality is the key to developing high-quality ubiquitous computing products. For this reason, various quality models have been defined, adopted and enhanced over the years, for example, the need for one recognised standard quality model (ISO/IEC 9126) is the result of a consensus for a software quality model on three levels: characteristics, sub-characteristics, and metrics. However, it is very much unlikely that this scheme will be directly applicable to ubiquitous computing environments which are considerably different to conventional software, trailing a big concern which is being given to reformulate existing methods, and especially to elaborate new assessment techniques for ubiquitous computing environments. This paper selects appropriate quality characteristics for the ubiquitous computing environment, which can be used as the quality target for both ubiquitous computing product evaluation processes ad development processes. Further, each of the quality characteristics has been expanded with evaluation questions and metrics, in some cases with measures. In addition, this quality model has been applied to the industrial setting of the ubiquitous computing environment. These have revealed that while the approach was sound, there are some parts to be more developed in the future.

1 Introduction

For decades, computing has centred about machines, not people. We have catered to expensive computers, pampering them in air-conditioned rooms or carrying them around with us. Purporting to serve us, they have actually forced us to serve them. They have been difficult to use. They have required us to interact with them on their terms, speaking their languages and manipulating their keyboards or mice. They have not been aware of our needs or even of whether we were in the room with them. Virtual reality only makes matters worse: with it, we do not simply serve computers, but also live in a reality they create.

On the contrary, in the future, computing environment should be human-centred. It will be freely available everywhere. It will enter the human world, handling our goals and needs and helping us to do more while doing less. We will not need to carry our own devices around with us. Instead, configurable generic devices, either handheld or embedded in the environment, will bring computing environment to us, whenever we need it and wherever we might be. As we interact with these "anonymous" devices, they will adopt our information personalities. They will respect our desires for privacy and security. We do not have to type, click, or learn new computer jargon. Instead, we will communicate naturally, using speech and gestures that describe our intent, and leave it to the computer to carry out our will.

[†] Email addresses: h.ryu@massey.ac.nz, g.y.hong@massey.ac.nz, h.a.james@massey.ac.nz

The idea of “ubiquitous” computing first arose from contemplating the place of today's computer in actual activities of everyday life. In particular, anthropological studies of work life (Suchman, 1987, Lave and Wenger, 1991) suggested that people primarily work in a world of shared situations and unexamined technological skills. However, the computer today is isolated and isolating from the overall situation, and fails to get out of the way of the work. In other words, rather than being a tool through which we work, and so which disappears from our awareness, the computer too often remains the focus of attention. And this is true throughout the domain of personal computing as currently implemented and discussed for the future.

Getting the computer out of the way is not easy. This is not a graphical user interface (GUI) problem, but is a property of the whole context of usage of the machine and the affordances of its physical properties: the keyboard, the weight and desktop position of screens, and so on. The problem is not one of "interface". For the same reason of context, this was not a multimedia problem, resulting from any particular deficiency in the ability to display certain kinds of real-time data or integrate them into applications. The challenge is to create a new kind of relationship of people to computers, one in which the computer would have to take the lead in becoming vastly better at getting out of the way so people could just go about their lives.

Therefore, ubiquitous computing systems are asking to bring together a number of research communities that have been focusing on various aspects of ubiquitous computing. While it is obvious that researchers from these various communities will need to co-ordinate and communicate to achieve the goal of ubiquitous computing it does not diminish the difficulty of the task.

We are proposing that one way to bring about a larger sense of community is through the development of evaluation frameworks. This includes participation of the researcher community in determining what to test, how to test, and appropriate metrics to use. This activity helps to focus the research community and also helps to show progress in the area. Focusing efforts on specific directions will be able to give the community direction and will be instrumental in demonstrations of success. As success is achieved in these focused areas, the community can evolve the evaluation in new directions, based on lessons learned[‡].

Ubiquitous computing poses more complex evaluation methodologies than the traditional desktop computing environment. However, there are several possibilities. One possibility is to start by conducting evaluations on the various aspects that make up subsystems of ubiquitous computing: perceptual user interfaces, dynamic service discovery, wireless networking services, distributed data systems and input and output using distributed user interfaces. Issues here would be to establish metrics and evaluation methodologies for individual components and to determine what, if anything, successful evaluations imply about the entire system.

Another possibility would be to evaluate individual systems, end-to-end, as they are built using traditional usability evaluation methodologies. This would give us information about individual systems and would perhaps allow researchers more flexibility in choosing particular domains. Would this necessitate changes in usability evaluation methodologies? Typical usability metrics are effectiveness, efficiency, and

[‡] www.nist.gov

satisfaction. Are these valid metrics for technology that is still under development? Is it feasible to compare systems across domains of use?
Actually, the main research purpose of the project is to identify various evaluation scopes of the ubiquitous system, so we will take the first approach to provide what evaluation aspects should be able to consider in the ubiquitous system.
For this end, as a first step, Chapter 2 surveys the current state of ubiquitous systems in academic or industrial end, caricaturing our aims of this research. As a subsequent investigation, the components of the ubiquitous systems are considered in Chapter 3. Building on this, Chapter 4 discusses what evaluation scopes and measurement must be taken in the evaluation process. Finally, we will provide a brief case study to apply our framework proposed in this report for the industrial setting.

2. Current State of Ubiquitous Systems: lessons from two examples

2.1 Mobilised university: Seton Hall University examples

One of the most successful ubiquitous computing environments can be found in the teritary education sector. Ubiquitous computing, in this case, refers to educational programs aimed at making a computer available to all members of a learning community. Thus, the goal of this ubiquitous computing environment is to ensure that everyone in the learning community has access to the necessary learning materials (many of which are computer- or network-based) at “any time, any place.” Practically, these programs usually involve ensuring that all students have some form of laptop computer and access to the Internet.
The concept of ubiquitous educational program has been around for more than a decade. These programs are heavily promoted by computer vendors, initially Apple™ Computer, but more recently by IBM™ and Compaq™. In 1999, IBM identified 36 colleges and universities when implementing its ThinkPad™ University program[§]. A study carried out in 1998 by Valley City State University in North Dakota identified some 84 campuses with ubiquitous computing programs, with several hundred more actively planning implementations.
For further discussion, we note a ubiquitous education program in Seton Hall University, which formed a strategic alliance with IBM Corporation to implement the University’s vision for the use of technology in support of its mission. As a result of the technology planning and the alliance with IBM Corporation, the University was ranked the 16th most wired campus in the 1999 Yahoo™ Internet Life survey of the 100 most wired colleges. More recently, Seton Hall University was awarded the 1999 Educause Leadership Award for Networking Excellence.
Seton Hall’s technology plan was developed as part of the University’s overall strategic plan in 1995. The technology plan was developed by a cross-functional team of faculty and administrators sponsored by the University’s chief academic and financial officers. The planning team received extensive feedback and support from the University community. This support included significant release time to work on the plan. The planning team was assisted in their efforts by a technology planning consultant from IBM Corporation. The University’s overall technology strategy is to develop and implement a learner-centred, network-enabled, distributed learning environment with a robust set of digital information resources in support of teaching, learning, scholarship,

[§] See http://www.ashe.ws/reviews/kinser.pdf for more details and the administration of the University. Critical to the success of this ubiquitous educational program focus on the use of technology to better serve teaching and learning.

One of the cornerstones of the University's strategic technology plan is known as Mobile Computing. Seton Hall's Mobile Computing Program is primarily viewed as an academic program combining ubiquitous access to technology, integration of information technology into the curriculum, and infrastructure and services that enable the effective use of information technology in teaching and learning. As part of this program the University licenses to each incoming undergraduate student the use of a current IBM ThinkPad computer, preloaded with a variety of instructional and professional software. Preliminary results indicate that the University's Mobile Computing Program has had a significant positive impact on indicators of effective learning, such as time on task, communication with the instructor, prompt feedback on assignments, and the like.

Seton Hall has made a major investment in the technology infrastructure and support services upon which the University's teaching, learning, and technology initiatives rest. The University's halls of residence are fully wired with voice and data connections for each student. Approximately one-third of the classrooms are "mobile ready," with multimedia projection facilities and data and power connections to each desktop. All classrooms have Internet connections available at the instructor's station. The University also has wired a number of public spaces on campus, including study carrels in the University Library, a local Coffee Shop, study lounges in the academic buildings – even the benches on the University Green.

Of course, however, such change has not come without challenges. The University has made a very significant commitment of its financial and human resources to this effort, and this has come as a result of both increases in tuition and fees as well as internal reallocation of resources. Despite the clear advantages of standardisation of technology and centralisation of support services, the difficulty (and cost) of providing the necessary infrastructure and support have been greater than anticipated. Nevertheless, we have found the successes, particularly the enhancement of the learning environment and the qualitative improvements in the campus culture, have outweighed the problems encountered.

2.2. Comfortable home: Microsoft EasyLiving Project

In addition to the educational sector, the business application is, of course, the major area of the future ubiquitous computing environment. In this context, at Microsoft™, the Ubiquitous Computing Group has created a system called EasyLiving that is a prototype of architecture and technologies for ubiquitous computing.

It consists of a living room, with couches, coffee tables, a number of desktop PCs, a large wall screen, TV, phones, etc. The room has a set of stereo camera heads mounted around the walls of the room. The cameras act as sensors that can detect the position of anyone in the room, their orientation and their direction of travel. The cameras can also detect whether a wireless keyboard is currently in the room, and who is currently using it. Because the room is able to detect the presence of someone in the room, if someone enters the room the lights automatically turn on.

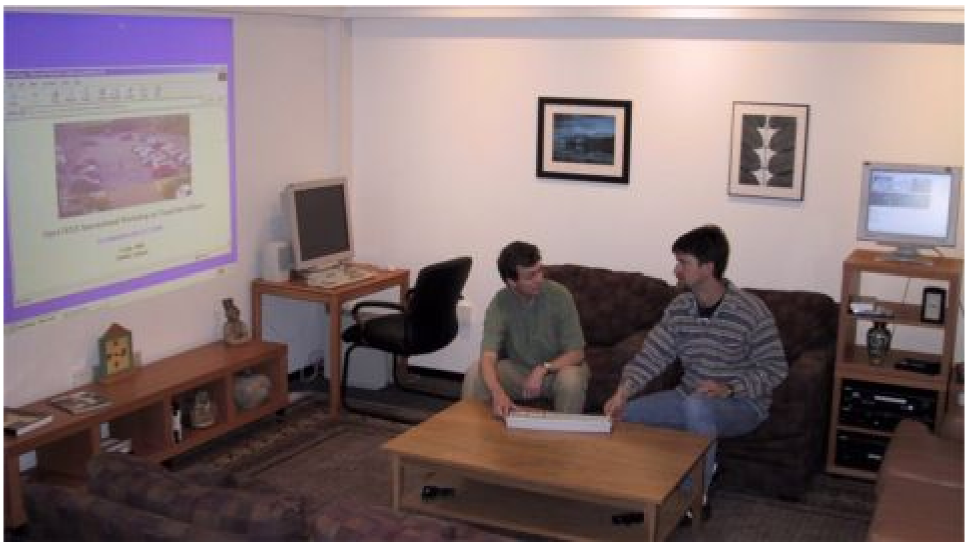


Figure 1. EasyLiving Room (Excerpt from Microsoft™ official website)

Before they enter the room, each person is given a personal identification number. If they identify themselves by logging into a PC in the room or touching a thumb scanner in the room the system determines who they are. They then have the ability to call up their personal desktop onto any computer screen in the room. If they then move to another computer, the person-tracker keeps track of their movement so that the new computer will automatically know who they are as well, without them having to log in again.

In the EasyLiving room, the display, keyboard, and mouse do not have to be attached to the same computer. Any keyboard and mouse can be used with any display. The software running believes that the user has 'a computer', and EasyLiving transmits the input and output across the network as necessary to use the devices indicated. Therefore, if you bring a laptop into the EasyLiving Lab, you can tell the system to move the display up onto the big wall screen. You continue to use the keyboard and pointer on your laptop. In this way, if you go to a meeting, you can show your presentation on the wall screen without having to hook up any cables or move your file onto the 'room' computer that controls the screen. This behaviour is known as disaggregated computing.

The EasyLiving system demonstrates many of the capabilities of ubiquitous computing, including mobile, wireless computing with migratory programs; an intelligent environment with context-awareness and location-sensitive computing; and disaggregated computing.

In summary, the concept of the ubiquitous computing environment is reliant upon many differing levels of system granularity, from systems software to presentation and session management on heterogeneous hardware. The following chapters discuss how these various levels of components have been dealt with the development of the current ubiquitous systems.

3 Challenges and Approaches of Ubiquitous Computing Systems

At present when we talk about computing, we generally think of a PC or workstation sitting on a desk, or increasingly a laptop or Personal Digital Assistant (PDA). However, Ubiquitous computing goes beyond this. Rather than a single computing device, devices surround the user. When Mark Weiser coined the phrase 'ubiquitous computing' in 1988 he envisioned computers embedded in walls, in tabletops, and in everyday objects. Computing devices surround us today in the form of microwaves, CD players, watches, electronic games, as well as computers and PDAs. The element that keeps this current environment from qualifying as ubiquitous computing is, of course, *connectivity*. In the future all the electronic devices in our surrounding environment could seamlessly communicate with one another in meaningful ways. For example the fridge could communicate with a digital shopping list, and add the required number of litres of milk that are needed for the following two days. In ubiquitous computing, a person might interact with hundreds of computers at a time, each invisibly embedded in the environment and wirelessly communicating with each other.

In order to meet the promises of the ubiquitous computing we briefly discussed above, it can be broken down into two major components, *mobile computing* and *intelligent environments*. Mobile computing has several aspects as follows:

- *Mobile Computers:* Making the computer and interfaces devices small, providing longer-lasting power and intelligent power management, making user interfaces that adapt to the size of the device and inventing new add-on devices for mobile computers.
- *Wearable Computers:* A special case of mobile computers designed for mostly hands-free operation by using passive sensors (cameras and microphones) and head-mounted displays.
- *Wireless Connectivity:* Providing continuous wireless connectivity to the network (through Bluetooth, Wireless IP (e.g., IEEE 802.11), GSM, GPRS, RFID) and maintaining service as the device is carried across service beacon boundaries while balancing the quality of service with the available bandwidth.
- *Migratory Computing:* Enabling applications to continue working while the device is moving between network beacons, especially through the migration of code between the mobile device and fixed computers, or from one fixed computer to another.

In contrast, an intelligent environment can be constructed through the combination of several elements as follows:

- *Mostly Hands-Free User Interfaces:* Technologies such as speech recognition, large displays, and visual user interfaces, which together allow the user to interact while physically distant from the devices. With these technologies, the devices can be mounted permanently in the room, while the person moves around and continuously interacts with them.
- *Context-Aware Computing:* In addition to explicit user interaction, the environment may have sensors that detect what is happening and what the user is doing in general terms. If this information is represented somehow and made available for applications to query, then those applications are context aware.
- *Intelligent Environment:* When a space has a representation of context, it may also have automatic behaviours that are triggered when things happen in the world, even without any explicit instruction from the user. If the only way

things happen is through such automatic behaviours, it may be called 'invisible computing'.

Each of these elements is reviewed in Chapter 4 together with the corresponding quality characteristics. The following sections discuss how each of the components has been developed for the promises of ubiquitous computing environment.

3.1 Post-PC era: Personal Physical Devices

Generally speaking, ubiquitous computing forces the computer to live explicitly in the world as an obvious physical entity with people. One of the main barriers of the user's uptake of this new paradigm is how much we can get the right information at the right time at convenience.

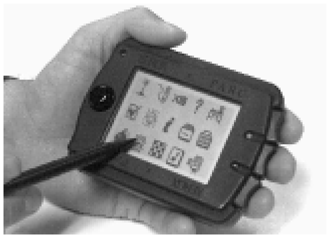


Figure 2. ParcTab™ (available from http://www.ubiq.com/parctab/pics.html)

For instance, a modern handset offers far more capabilities than early ubiquitous computing devices such as ParcTab in roughly the same form factor (see Figure 2). A typical phone today might include simple PDA applications such as a calendar and to-do lists, and even voice-recognition systems.

Modern mobile phones have another important property that Weiser associated with ubiquitous computing devices: many users view them as a commodity that can find and use a network anywhere they travel. GSM systems approximate this model of operating using a SIM card. By inserting their SIM card into a handset or some other equipment, people can automatically use the equipment, placing and receiving calls as if it were their own phone – they can even access their stored telephone numbers and text/picture messages. To many users, this demonstrated the end-system is becoming less important than the access it provides to the digital world.

3.2 Covert-monitoring era: Sensors

The sensor is one of the most important components of a ubiquitous computing environment. Recent advances in wireless communications and electronics have enabled the development of low-cost sensor network.

For example, in Japan, some elderly-people caring facilities are using a Laser Beam Sensor (LBS) to monitor elderly-people's movement. The LBS consists of separate transmitter and receiver units that project a wide beam of laser light between them. The portion of the laser light that reaches the receiver depends on the dimensions of the object (or portion of it), in the light beam. Measurement is achieved by comparing the amount of light received versus the amount of light transmitted.

Bluetooth is an emerging standard and a specification for small –form-factor, low-cost, short-range radio link between mobile PCs, mobile phones and other portable devices. In other words, it is a wireless connection between PCs, peripherals, and portables that will let devices share and synchronise information without having to make a physical

connection. Although the idea behind Bluetooth (wireless communication between devices) has been around for a while, it is the momentum behind this standard and the agreement among hundreds of vendors and manufacturers that has brought it to the verge of becoming a reality.
A Bluetooth network (known as Piconet) can allow the interconnection of eight devices in a radius of 10 metres. This network can be fixed or ad-hoc (a mobile or transitory network). In a Piconet, the master seeks the devices in its entourage by emitting requests (broadcast). The slave answers with its identification number. As many as 10 Piconets can overlap to form a Scatternet, linking up to 80 Bluetooth appliances. Beyond this, the network saturates. Indeed, only 79 transmission channels are employed by the Bluetooth protocol, a limit based on the frequency.
By default, Piconets can transmit up to 10 metres (about 30 feet). However, it can be increased upto 100 metres by increasing the power output from the 1mW (1 milliwatt) default to 100 mW. However, compared to GSM (Global System for Mobile communications), which consumes between 1.5 and 2 Watts, this is still a weak signal. Manufacturers are working to make Bluetooth devices that adapt to the necessary proximity, so as not to consume more energy than is necessary.
Bluetooth is not designed to compete with wireless local area networks. Even its close-range throughput of 1 Mbps doesn't compare with the 11 Mbps that the emerging standard for wireless LANs, IEEE 802.11, offers.
Instead, Bluetooth's promoters are positioning it as the technology for the Personal Area Network (PAN), and are targeting appliances that do not require large data throughput - like printers, personal computers, and mobile phones. One concept that has been put forward is the mobile PAN: a communication device clipped to the user’s belt could contain a GSM transceiver that communicates with the wider world. Meanwhile, the same device could have a Bluetooth transceiver that communicates with the user’s headset (replacing the mobile phone), their PDA and their MP3 player, allowing all these devices to communicate with each other and the larger world.
Since it is not a very expensive technology, it can easily be placed in many devices. Also, Bluetooth does not require an access point, unlike the traditional radio operator networks. It is well suited for mobile devices, since it can join a local Piconet quickly, as soon as the two devices are in a sufficient perimeter. And unlike infrared networks (like two Palm computers beaming each other), Bluetooth does not require to align objects for them to communicate.

3.3 Intelligent living environment: Smart House

Microsoft’s EasyLiving project (see Section 2.2) is a developing prototypical architecture and technologies for building intelligent environments, including computer vision for person-tracking and visual user interaction, multiple sensor modalities combined, device-independent communication and data protocols.
A smart house is one in which technology is integrated to a level where the technology and appliances in the house help make life easier, safer and more enjoyable for the inhabitants. Much work has been done on individual technologies, but little has been done on their integration into a cohesive whole.
For a house to be considered as truly ‘smart’, it requires three things:

it needs to know who is in the house, where in the house they are, and what special needs or preferences they may have; it needs a central management system which, based on the occupants identities and locations, can co-ordinate all the smart appliances and devices in the house to best suit those occupants' needs. An effective input device, such as a speech recognition system that allows the users to communicate with the house in a natural manner, is crucial.
Beyond the smart house, HP Labs has been working at the intersection of nomadicity, appliances, networking, and the web. HP calls their vision of the future "Cooltown" - a vision of a technology future where people, places, and things are first class citizens of the connected world, wired and wireless - a place where e-services meet the physical world, where humans are mobile, devices and services are federated and context-aware, and everything has a web presence. In Cooltown, technology transforms human experience from consumer lifestyles to business processes by enabling mobility. Cooltown is infused with the energy of the online world, and web-based appliances and e-services give you what you need when and where you need it for work, play, life[**].
In a similar concept, a wired home cluster, for instance, Korea's Tower Palace apartment is one of the industrial examples of where a currently working smart house can be found in the living culture. As there are densely populated living areas, the connectivity can be easily set up. The designers of the apartment are intending to use ubiquitous network such as wireless, mobile, and broadband systems.
In summary, current state-of-the-art of smart living such as smart house or smart building is relatively easily applied to the implementation of a new smart house. However, the main aims of living in the smart house should be 'amenity'; this leads us a challenge such as how we can measure this.

3.4 Smart applications: Ubiquitous services

Recently, Vodafone UK and Telecom NZ have introduced a ubiquitous service to provide some location awareness service for mobile phone users. In addition, Japan's Zojirushi Corp., recently released the i-pot, an Internet-enabled hot pot that dispenses boiling water for tea. Hot pots are common, everyday items in Japanese homes. This hot pot, however, does more than this. In addition to boiling and dispensing water, i-pots send usage statistics to a website that tracks users' tea-drinking patterns. Caregivers can monitor a user's well-being by watching for breaks in their tea-drinking routine, which are indicated in twice-daily email reports or by checking a website. The target market for the i-pot is elderly people whose children or grandchildren might live too far away to monitor them directly[††]. The Metro[‡‡] supermarket in Germany uses a "check-out free service", an intelligent scale to automatically identify and weigh fruit and vegetables, electronic shelf labelling and other shopping aids such as an automatic self check-out machine and Radio Frequency Identification (RFID) technologies.

[**] http://www.cooltown.com/cooltown/index.asp
[††] http://www.mimamori.net/
[‡‡] http://www.future-store.org/servlet/PB/-s/joghep1ahqi8vdh5hi98tds5rnrrruw/menu/1002263_l2/1090604414107.html

3.5 A future ubiquitous computing environment: the Oxygen project

One of the major ubiquitous computing environments, which is currently being developed at MIT, is the Oxygen project. It enables pervasive, human-centred computing through a combination of specific user and system technologies. Oxygen's user technologies directly address human needs. Speech and vision technologies enable us to communicate with Oxygen as if we are interacting with another person, saving much time and effort. Automation, individualised knowledge access, and collaboration technologies help users perform a wide variety of tasks that they want to do in the ways they like to do them.
Oxygen's device, network, and software technologies dramatically extend the users' range by delivering user technologies to them at home, at work or on the go. Computational devices, called Enviro21s (E21s), embedded in users' homes, offices, and cars sense and affect their immediate environment. Handheld devices, called Handy21s (H21s), empower users to communicate and compute no matter where they are. Dynamic, self-configuring networks (N21s) help machines locate each other as well as the people, services, and resources users want to reach. Software that adapts to changes in the environment or in user requirements (O2S) help users do what they want when they want to do it.
The following section discuss the prototypes of current ubiquitous systems and the working models of each components of the ubiquitous systems.

3.6 Challenges and approaches of the ubiquitous computing environment

As we discussed above, many advances on ubiquitous computing systems have been made on to accomplish the promises of the ubiquitous computing environment. While many of them have made a major breakthrough on some challenges in terms of the technological aspects, there is still a highly integrated environment required. The following sections review what research challenges must be dealt with in the near future and how they are currently being tackled. This will direct us to crucial quality characteristics of the ubiquitous system in the various levels and scopes, which is discussed in Chapter 4.

3.6.1 Ad-hoc network

Ad-hoc networks are multi-hop wireless networks where nodes may be mobile. These types of networks are used in situations where temporary network connectivity is needed. Ad-hoc networks are formed on a dynamic basis, i.e., a number of users may wish to exchange information and services between each other on an ad-hoc basis.
An example of this may be found in a disaster relief situation. Here an ad-hoc network could enable emergency services to co-ordinate emergency services more effectively or enable medics in the field to retrieve patient history from hospital databases (assuming that one or more of the nodes in the ad-hoc network has connectivity to the Internet).
Ubiquitous computing environments can be defined as environments that allow people to perform tasks efficiently by offering unprecedented levels of access to information and assistance from computers. In this context, ad-hoc networks will play a significant part in these environments, allowing people to exchange information and services; for example, people at a meeting could create an ad-hoc network using their PDAs or

Laptops and exchange information relevant to the meeting. Indeed there are endless examples of where their use could be found.
A ubiquitous network is one that is present everywhere throughout an area and feature the ability to detect where users are within the network and based on that information meet their needs. The following challenges have been posed in the literature, and they will be further discussed in Chapter 4:

- Fault tolerance
- Scalability
- Flexibility
- High-sensing fidelity
- Error control

3.6.2 Software technologies for the ubiquitous system

Few ubiquitous computing environments that exist today tend to be highly specialised and based on application-specific software. Applications developed for interactive environments should be able to interconnect and manage large numbers of disparate hardware and software components. They should operate in real-time; dynamically add and remove components to a running system without interrupting its operation; control allocation of resources; and provide a means to capture persistent state information. Frequently these components are not designed to cooperate, so not only do they have to be connected, but also there is a need to express the "logic" of this interconnection. In other words, inter-component connections are not merely protocols, but also contain the explicit knowledge of how to use these protocols. Thus, viewing the connections simply as an application-programming interface is not enough. Cooperation among different applications is also difficult to achieve without a common platform. In order to model applications in this domain there is an important need to define a common design methodology based on new paradigms independent from the technology. A model to abstract the main components of a ubiquitous computing environment is needed in order to formalize the development of interactive environment applications. These components may be classified into three abstraction layers (Maffioletti, 2001):

- Physical deals with technological constrains.
- Middleware defines structure and the cooperation of abstract services.
- Application concerns the user interfaces

Thanks to these abstractions a middleware will present a uniform access abstraction across different ubiquitous devices, allowing them to interact and co-operate. This will allow the writing of applications scaling both on services offered, and on devices composing the system. The model is intended to provide a standardised view of basic interactive environment functionality. The main requirements that must be provided by a middleware infrastructure for services in ubiquitous computing would fall under the following headings:

Mobility

New wireless communication technologies provide connectivity for laptop computers and PDAs; phones and organisers offer the processing power, and application-level

protocols such as the wireless application protocol—a tiny first step—and allow convenient applications to run on these devices. Undoubtedly, these developments point toward the widely expressed goal of accessing and processing information almost “anywhere and any time.”
Independent of a person’s current location, one can already use voice communication via mobile phones almost anywhere, and have worldwide access to personal e-mail, bank accounts, and home-country news over the World-Wide Web. Mobility introduces a key technical challenge because the available resources vary widely and unpredictably.
Communication bandwidth and error rates change dynamically in wireless communication networks, a mobile system’s battery power gets consumed, portable devices can be temporarily switched off or unreachable because of network partitions, and the monetary cost of communication can vary significantly. Envisaging a middleware system that makes these dynamics completely transparent is difficult; so middleware must support the applications to explicitly accommodate these changes.
Another mobile-computing issue, location awareness, demands that mobile-computer applications know their operating environment for context-dependent activities, such as giving directions or employing more or less stringent security mechanisms. The characteristics of Message Oriented Middleware (MOM) would be suitable in a ubiquitous environment. MOM does not require constant connections to be held between communicating devices, as it can pass messages asynchronously, accounting for ad-hoc connections in the ubiquitous environment.

Heterogeneity

The ubiquitous computing vision assumes that future computing will comprise diverse computing devices ranging from large computers to microscopic, invisible processing units contained in objects that are used in daily life (e.g., light switches, coffee mugs, friges, etc). These computing devices may all need to communicate over different network mediums and protocols, (e.g., IEEE 802.11b, Bluetooth, IrDA, etc.).
Present middleware technologies such as CORBA can be implemented in different languages, allowing heterogeneous ORBs talk to each other. Other OOM technologies such as SOAP communicate in a structured format (XML) over common network protocols (HTTP). Message oriented middleware can also provide data communication over diverse networks, such is the case for Java Messaging Service, where as long as the applications communicating are running in a Java Virtual Machine (JVM) corresponding to their platform, communication is possible.
Middleware for a ubiquitous environment must follow the characteristics of present middleware technologies in regards to the way they can be implemented in heterogeneous environments, and communicate over heterogeneous networks with dissimilar protocols.

Scalability

The middleware has to scale well for both a large number of co-operating services, which realise the application in each application context, and a large number of devices

involved each time applications are used. Services represent the logical dimension of the application, while devices represent its physical dimension.
Different middleware technologies scale in different ways. OOM such as CORBA would be considered a connection oriented middleware; this is to say if two or more ORBs are to communicate, there must be an open connection between them. Middleware must provide applications with consistent functionality the more a system scales. In a ubiquitous environment the amount of connections between devices at any time could be immense. The management of these connections could require more resources than available, which is not desirable in such a resource-restricted environment.
Message oriented middleware (MOM) mostly works on a connectionless basis, where the sender and receiver may be disconnected when communication is initiated. When a system, using MOM, is scaled to an environment with possibly hundreds of devices communicating with each other, such as a ubiquitous environment, messages being passed may build up (and have to be stored somewhere before being delivered) because of devices being disconnected for unpredictable amounts of time. This may compromise the performance of critical systems running in a ubiquitous environment.
A possible solution to the scalability issue for ubiquitous environments is to group small areas together, such as a room. In this room, because the devices in the room may be made up of static and dynamically connected devices, it would be safe to assume a limited number of devices may be in operation at any one time. Using these assumptions, different middleware solutions may be used in combination with each other in order to control the scale of the environment. For example in a small-scale environment, an OOM may be used to control connections between devices, such as the .NET framework, which is based on XML Web services communicating through basic Web protocols. A collection of these environments could be considered another group communicating at a higher level, connecting these environments together. Using a method like this would break down the environment, into a controllable, scaling, environment.

Configurable / Re-configurable

It is becoming increasingly apparent that most existing middleware technologies cannot accommodate the great diversity of application demands in areas like mobile and ubiquitous computing. The main reason for this is the black-box philosophy adopted by existing middleware platforms. In particular, existing platforms offer a fixed service to their users and it is not possible to view or alter the implementation of this service—that is, they are closed systems. This basically means that all network and connection information is hidden from the application layer, thus not allowing applications see changes in network topologies. Applications should be able to be configured to any network topology (IEEE 802.11*x*, GPRS, Bluetooth, etc), thus the underlying middleware should not be considered as a black box paradigm but more of a book that can show certain required information about the underlying network topology to the application layer.
Devices in ubiquitous environments will automatically detect other devices; forming ad-hoc networks spontaneously such as when using Bluetooth technology. A

middleware infrastructure for a ubiquitous environment will have to re-configure itself to changes in the network, and also be able to reflect this change at the application layer. The Java-based Jini™ system appears to anticipate these requirements. Jini defines a middleware infrastructure for spontaneous networking in which Java objects can discover, join, and interact with communities of devices. Whether the Jini approach will scale to the requirements of ubiquitous computing, however, remains unclear.

Quality of Service

Increasing concerns about service quality have led to several proposals that advocate integrating Quality of Service (QoS) management into networking and distribution infrastructures. QoS management at the middleware and application levels aims to control attributes such as response time, availability, data accuracy, consistency, and security level.

From the perspective of the Internet, QoS concerns seem to arise automatically when commercial applications meet a best-effort communication environment. When clients must pay for a service, they are certainly concerned about QoS, and they will expect to pay less for lower quality. From a ubiquitous computing perspective, the guarantee of QoS is an even greater issue. For example, if a heart monitor was part of a ubiquitous environment, then the guarantee of the quality and delivery of this important data could mean life or death. Also for public acceptance of a ubiquitous environment, people must be able to trust the security of personal data that can under-go potential total monitoring (Geihs, 2001)

The ubiquitous computing environment introduces extra issues regarding QoS that present QoS architecture for the Internet or wired networks do not cover. This new environment introduces problems such at wireless channel fading and mobility (Naghshineh and Willebeek-LeMair, 1997). These are two very important elements that have to be addressed if QoS architecture is to be successfully mapped to a ubiquitous computing environment. As these problems can affect the performance of a network and application, it is often proposed that QoS be handled at the middleware layer (Nahrstedt, 2001), suggesting that QoS-aware middleware architecture for ubiquitous computing environments have to address four main aspects of QoS:

First, the QoS specification must allow the description of application behaviour and QoS parameters; second, QoS translation and compilation to translate specified application behaviour into candidate application configurations for different resource conditions; third, QoS setup to appropriately select and instantiate a particular configuration; finally, QoS adaptation to adapt to runtime resource fluctuations. Under these four aspects the problems that the ubiquitous environment introduces to QoS may be tackled.

An example of a problem introduced be ubiquitous environments would be the transmission and receipt of multimedia data to and from a wireless media player application. Applications such as these can run using different configurations, e.g., streaming videos using ISO 's MPEG layer 2 (ISO/IEC, 1994) encoding or ITU 's H.261 (ITU-T, 1993) encoding. The former method is for high quality video storage and transmission, which would require a large amount of network resources to be available. The latter is used for low bit-rate videoconferencing over narrowband integrated services digital network lines (ISDN) and telephone lines. In a ubiquitous computing environment due to fading signals, network bandwidth may drop considerably, in a non-linear rate. In this event a QoS aware middleware infrastructure should be able to recognize that a change in network performance has occurred, notice that the application can handle different configurations, and adjust the video encoding method to suit. In this example, it is clear that the application layer needs to know some information about the network layer and visa versa. QoS aware middleware should provide an abstract link between the two layers, controlling the application when network connectivity fluctuates. Another issue that should be considered is handing off between two networks. A hand-off is where a mobile device moves from one network to another, vertical hand-off being when a device moved from one network topology to the next (Wi-Fi to GPRS), and horizontal-hand off being roaming between similar network topologies (Wi-Fi to Wi-Fi). QoS aware middleware should be able to adjust applications to react to any changes in network resources.

In ubiquitous computing environments, the question of guaranteed QoS is still open. Present technologies that provide some forms of Quality of Service, e.g., Nahrstedt (2000), would include middleware infrastructures that provide a method of application configuration and adaptation to environment changes through QoS programming environments. In a ubiquitous computing environment, containing a wide variety of devices, network topologies and applications, it would be necessary to have custom designed QoS resolutions to suite different configuration changes applications may need to undergo in the event of fading or mobility (Nahrstedt, 2001). This implies that the most suitable application service paradigm for ubiquitous computing is adaptive in nature; hence the provision of this adaptive quality may be provided at the middleware layer (Lauff and Gellersen, 2000).

Jini is one of the widely using middleware for the network technology, which includes JavaSpaces Technology and Jini extensible remote invocation (Jini ERI), is an open architecture that enables developers to create network-centric services - whether implemented in hardware or software - that are highly adaptive to change. Jini technology can be used to build adaptive networks that are scalable, evolvable and flexible as typically required in dynamic computing environments. Jini that runs on top of Java allows services to dynamically join and exit without impact on the network or the network users. As a service based architecture, Jini provides a solution for the evolving ubiquitous computing requirements.

In addition, Jini is a technology that will allow developers and manufactures create a range of computerised devices that can instantly connect both wired and wireless into a network to share services regardless of the underlying operating system or hardware. Jini software gives network devices self-configuration and self-management capabilities; it lets devices communicate immediately on a network without human intervention.

In contrast, some other technologies are competing with Jini such as UPnP, HP JetSend. The following headings briefly review them.

Universal Plug and Play (UPnP)

Microsoft's Universal Plug and Play (UPnP) is open distributed networking architecture for pervasive, peer-to-peer connectivity of intelligent devices. UPnP supports pervasive,

invisible networking, zero configuration, and dynamic formation of device communities.
UPnP is built using established open Internet standards such as IP, UDP, TCP, HTTP, XML, and SOAP, for various devices to discover and join a network dynamically. No configurations and no device drivers are required. UPnP internetworking is based on IP standards. Thus, every device joining the UPnP community acquires a network address of its own. UPnP is platform and language independent and is geared toward dynamic ad-hoc networking of devices, however it does not handle services provided by software implementation or any other entity. As such it is a system for ad-hoc networking but does not address the requirements of large scale service deployment. Jini works both at the ad-hoc and enterprise level.

HP JetSend

HP™ JetSend is a device-to device communication technology that allows devices to negotiate information exchange intelligently; JetSend enables devices to transfer data without needing to know anything about the other device. JetSend is both complementary to and competitive with Jini.
JetSend can work with both wired and wireless protocols. This, however, assumes the existence of a JetSend-enabled device at both ends of the network. The communication model does not require driver installation or configuration, which makes it easy to plug and play with different devices. JetSend does not depend on a transport protocol, hence it can work with any bi-directional transport protocol, including TCP/IP, IR, RF, Bluetooth, IEEE1394, and others. JetSend can work with various networking technologies and can coexist with such technologies as UPnP, and Chai ApnP.
Since Jini is protocol independent, it can embrace JetSend as a communication protocol for communication between devices. JetSend, is targeting a specific requirement, that of remote printing, using JetSend as the communication technology. While this is an initial starting point, it is limiting for other service/communication requirements.

HP Chai

HP™ Chai is a full-featured Java based appliance platform for developing and running applications on embedded devices. While being implemented in Java, the ChaiAppliance communications mechanisms are based on Web standards such as HTTP and XML, rather than using Java's primitives. Chai Appliance has been developed using the Chai Virtual Machine, HP's clean-room developed version of Java. This could possibly make Chai Appliance Plug and Play useful as a bridge between Jini and UPnP.

3.6.3. Physical artefacts

The approach that Weiser (1991) took was to attempt the definition and construction of new computing artefacts for use in everyday life. The most ubiquitous current informational technology embodied in traditional artefacts is the use of written symbols, primarily words, but including also pictographs, clocks, and other sorts of symbolic communication. Rather than attempting to reproduce these objects inside the virtual computer world, leading to another "desktop model" [Buxton 90], instead Weiser wanted to put the new kind of computer also out in this world of concrete information conveyers. And because these written artefacts occur in many different sizes and shapes, with many different affordances, so Weiser wanted the computer embodiments to be of many sizes and shapes, including tiny inexpensive ones that could bring computing to everyone.

The physical affordances in the world come in all sizes and shapes; for practical reasons our ubiquitous computing work begins with just three different sizes of devices. The first size is the wall-sized interactive surface, analogous to the office whiteboard or the home magnet-covered refrigerator or bulletin board. The second size is the notepad, envisioned not as a personal computer but as analogous to scrap paper to be grabbed and used easily, with many in use by a person at once. The cluttered office desk or messy front hall table are real-life examples. Finally, the third size is the tiny computer, analogous to tiny individual notes or PostIts™, and also like the tiny little displays of words found on book spines, lightswitches, and hallways.

One of the common physical devices which would be considered at the current state of the development is the mobile phone. Today all mobile phones come with the features that are standard across all brands and models of phones: the ability to make clear voice communication, to store names and numbers, to send and receive SMS (Short Messaging Service), to customise ringtones, logos, etc and to be able to roam across networks/countries etc. But now phones are moving away from simply providing the basic features detailed above. The advances of 2.5G networks and the start of the implementation of 3G ones, haven given rise to the development of phones that take advantage of these new capabilities. Many phones are now data enabled, allowing them to make use of GPRS and 3G in order to send data across the same network they make calls on. They can send and receive files, e-mail, multimedia (including multi-media messaging service (MMS)), etc. Because of the ability to send multimedia such as sound, video and still images, the next generation of mobiles will not feature displays suited for simple lines of text but ones capable of viewing the media properly. Already Samsung, Sony Ericsson and Nokia have phones that feature colour displays capable of displaying multimedia.

Figure 3. Samsung, Sony Ericsson and Nokia phones featuring colour screens

Some phones, such as the Sony Ericsson T68i, allow the user to attach a camera to the phone, take images and then send them to either another multi-media messaging service capable phone or by e-mail to any PC.

Figure 4. Sony Ericsson Capture Cam

Nokia released the 3410, using Java™ technology, the user can further enhance their phone with interactive, dynamic games, downloaded and stored in the phone for use. And when they are no longer needed, it is easy to delete them. Examples of downloadable Java™ applications include interactive games, life management tools, travel-related applications and information tools. The possibilities are virtually limitless, and hundreds of thousands of developers around the globe are working to realise potential of Java in the wireless world.

Palmtops are computers that are designed fit in your palm. They allow the user to carry out many of the tasks that could be performed on a full size desktop PC. These include word processing, e-mail, calendars, databases, image processing, web browsing, etc. Some of the available functions might be scaled down due to various restrictions, but usually any data created on the palmtop can be transferred to the desktop PC and vice versa.

Palmtops are usually broken down into two groups according to which operation system they run, either the Palm OS or Microsoft's Pocket PC (a reduced version of Windows for palmtops). Many companies manufacture palmtops under license from the Palm Corporation but while they trade under different names they operate the same as the Palm made by the Palm Corporation.

Originally called the Palm Pilot the latest models of the Palm are the M500 and M515. The M500 has a monochrome display while the M515 has a colour display. The Palm uses Palm operating system (Palm OS), has a 33Mhz processor and 16Mb RAM. It provides the user with the basic functions of any PDA, address book, calendar, calculator, memopad, etc. But it will not run the same applications that are run on a Windows machine. So porting between the two is a bit difficult. With the inclusion of a Java VM (Virtual Machine) it is possible to run custom applications. The Palm is at the lower end of the palmtop market in terms of performance. But it is one of the leading brands currently with 40% of the market share. But it is losing ground to devices that use Pocket PC such as the iPaq™ devices currently made by Compaq.

Pocket PC's popularity can be linked to the fact that Windows is the OS of choice on desktop PC's and because Pocket PC functions like Windows in terms of both appearance and applications, users find it easier to use. There are many more

applications available for Pocket PC devices than for Palm ones, and the devices themselves tend to have more memory and faster processor. Again all the devices are very similar, regardless of their brand name, but the iPaqs from Compaq are the most popular.
The latest iPaqs from Compaq, the H3970 and H3950, both come with 400Mhz Intel processors 64Mb RAM and 48Mb ROM and full colour displays. The H3970 is also Bluetooth capable, which allows it to communicate with other Bluetooth capable devices. It is therefore easily expandable with a range of different devices, such as headsets, wireless keyboards, printers, etc.

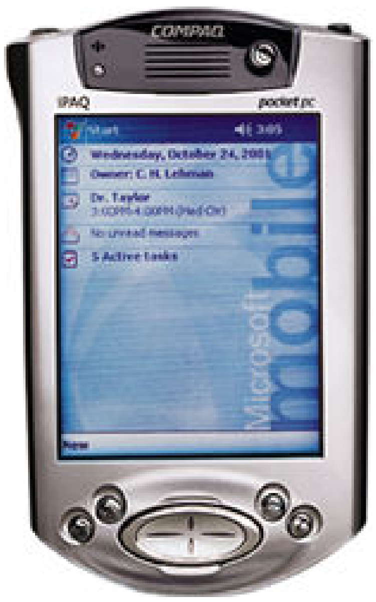


Figure 5. Compaq iPaq H3970

Smartphones are a hybrid of palmtops and mobile phones. They combine all the functions of the mobile phone with either all or a reduced set of the functions of a palmtop. At the very least they will allow the user to send and receive e-mails, store contact information, schedule appointments, word process, etc.
In fact many smartphones use Microsoft's Pocket PC or a Pocket PC Phone Edition, which has been recently launched by Microsoft. The Phone Edition adds phone related features including voice communication and SMS text messaging. An example of a smartphone is the xda from O^2 (although it is also available under other mobile network operator's brand names).

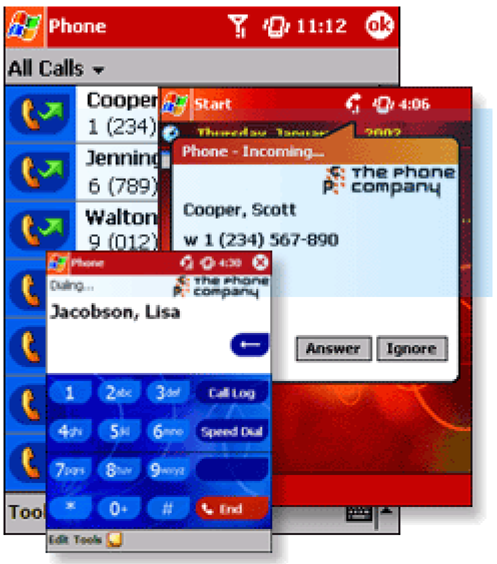


Figure 6. Microsoft Pocket PC Phone Edition Screenshots [§§]

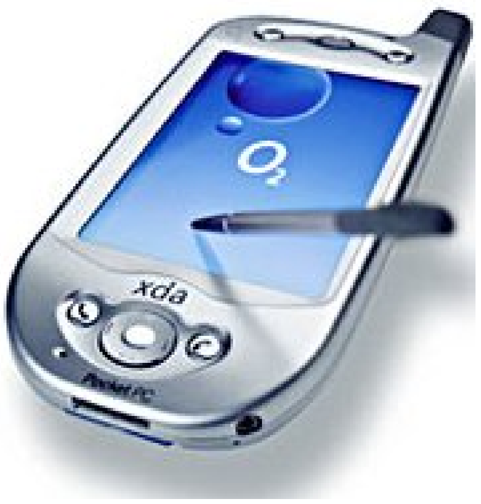


Figure 7. O^2 xda

It runs Pocket PC 2002 and allows the user not only to conduct voice and data communication (using GPRS) but also to e-mail, browse the web, write Word and Excel documents. The user enters text using the stylus on the touch sensitive screen and hand writing recognition software converts this into typed text. This is typical of all smartphones and palmtops.

[§§] Available from **http://www.microsoft.com/mobile/pocketpc/phoneedition/default.asp**.

Figure 8. T919

Other smartphones have additional features. The T919 from Austrian manufacturer Tel.me has an integrated digital camera. Other smartphones are Bluetooth enabled and are thus compatible with almost an unlimited range of devices and expansions.
Handhelds are portable computing devices that are bigger than a palmtop but smaller than laptops. They offer the same features as palmtops and in a few cases more. Their major difference is usually that they feature larger screens and use keyboards as well as styli to input data from the user.
The market leader is Psion. However handhelds are not as popular as they were before the introduction of palmtops, mainly due to the rise in popularity of palmtops that offer the same features but are smaller in size (more portable), as well as the fact that many of the palmtops run Windows. Psion uses an operating system (OS) called Symbian OS, developed by Symbian, which was established as a private independent company in 1998 and is owned by Nokia, Matsushita (Panasonic), Motorola, Psion, Siemens and Sony Ericsson.

Figure 9. Nokia Communicator

The OS is now being geared up towards use in smartphones, such as the Nokia Communicator. But once again Windows Pocket PC is gaining the majority of the market share.
A head mounted display (HMD) would be a possible physical device for the ubiquitous computing system, consisting of a headpiece that incorporates a display that replaces

the ‘real world’ view of the user with one that is projected onto the display. A head mounted display typically provides a stereo-optic (3D) view through the use of two LCD or small CRT displays. However the use of LCDs in HMD is becoming standard, due to the fact that they allow a much slimmer compact unit to be produced.

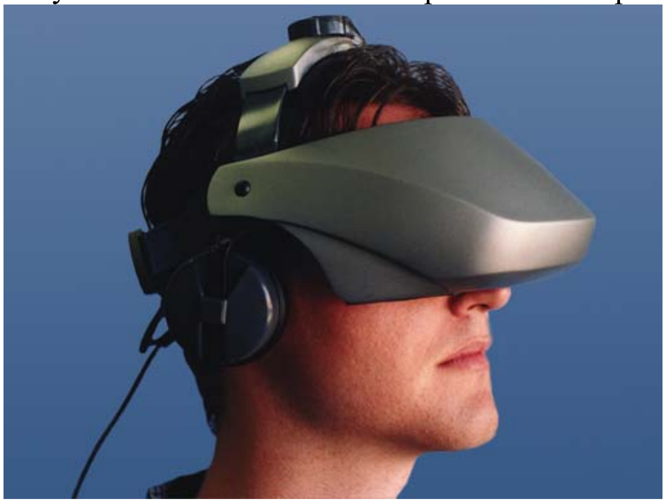


Figure 10. A example of HMD

As with any display device, the features, characteristics and price of HMDs can be quite varied. HMDs made for home use (typically as an add-on to an existing games console) cost about $500 and provide an adequate display for game playing in virtual reality situations. HMDs developed for the military or specialised training situations such as flight or surgical training are in the region of $20,000 but are capable of providing a near ‘real world’ image display.
A wearable computer is a computer that is subsumed into the personal space of the user, controlled by the user, and has both operational and interactional constancy (i.e., is always on and always accessible). Most notably, it is a device that is always with the user, and into which the user can always input data and execute a set commands, and in which the user can do so while walking around or doing other activities. The most salient aspect of computers, in general, (whether wearable or not) is their reconfigurability and their generality, e.g., that their function can be made to vary widely, depending on the instructions provided for program execution.

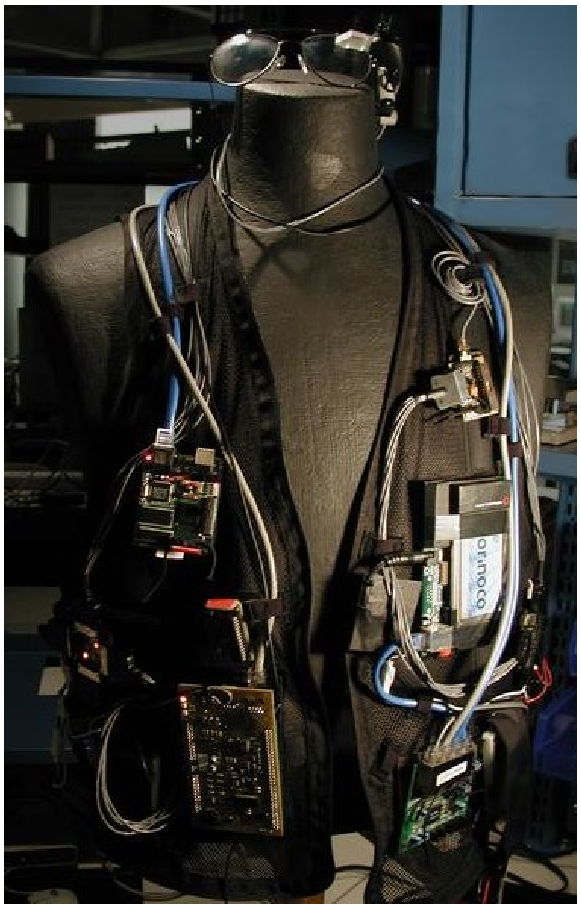


Figure 11. An example of wearable computer

With the wearable computer this is no exception. The wearable computer is more than just a wristwatch or regular eyeglasses: it has the full functionality of a computer system but in addition to being a fully featured computer, it is also connected to the wearer. This is what sets a wearable computer apart from other wearable devices such as wristwatches, regular eyeglasses, wearable radios, etc. Unlike these other wearable devices that are not programmable, the wearable computer is as re-configurable as the familiar desktop or mainframe computer.

Another application is the BlindVision system developed by Steve Mann at the University of Toronto. BlindVision is aimed at visually impaired people. It is a personalised radar system that is integrated in a close-fitting vest and which is able to process objects in the vicinity of the wearer. Reflected radar signals from nearby objects are transformed by a wearable computer and sent to the vest, which sends electric stimuli to the wearer. The exact position of a moving object with respect to the wearer is simulated, as well as the proximity: closer objects exhibit stronger "pressure" via stronger current, while objects further away accordingly output milder current. In a very real sense, one can experience what could be described as the sensory vision.

MEDIWEAR is closely related to BlindVision, but with an interesting twist: where BlindVision is involved with processing the outside stimuli and presenting them introspectively, MEDIWEAR does the opposite - clothes with embedded wearable computers closely monitor the wearer's body functions. The moment that any one of them becomes critical, the predefined medical unit is notified remotely.

The next step in mobile computing beyond wearable computing will ultimately be digestible computing: where the computing device is within the user and the interacts with the user using the user's physical signals, such as nerves, muscles, etc.
Some examples of this new area of mobile computing already exist. In February 2002 a family in Florida were the first humans to be implanted with the VeriChip, developed by Applied Digital Solutions. An implantable, 12mm by 2.1mm radio frequency device, VeriChip is about the size of the point of a typical ballpoint pen. It contains a unique verification number. Utilising an external scanner, radio frequency energy passes through the skin energising the dormant VeriChip, which then emits a radio frequency signal containing the verification number. The number is displayed by the scanner and can be used to access remote databases with information about implants, the person's health history and other relevant information. The database provides people, medical providers and manufacturers with a rapid, secure and non-invasive method of obtaining medical and medical device information. Future applications may include full medical record archival/retrieval for emergency medical care.

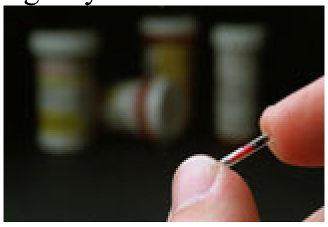
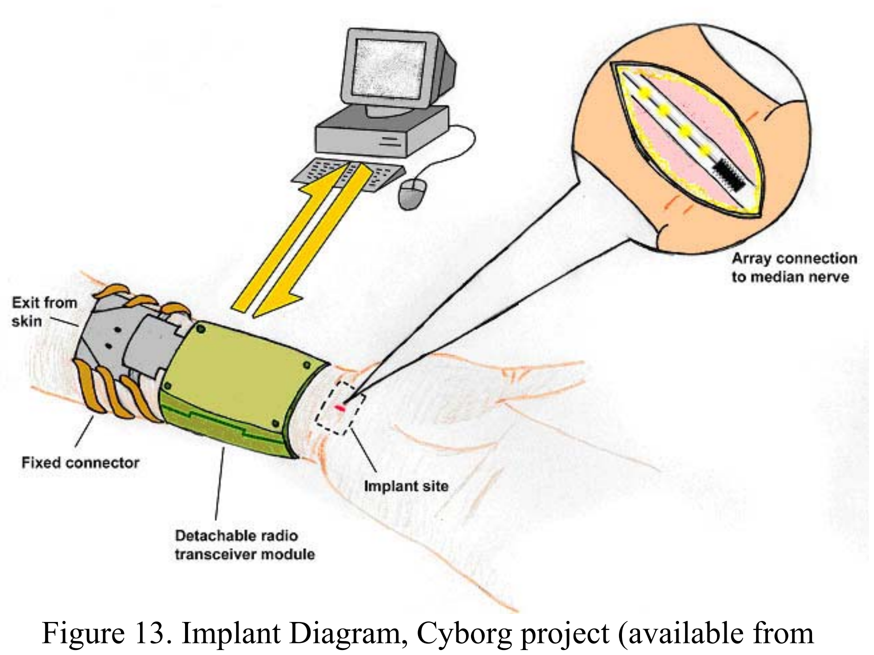


Figure 12. VeriChip from Applied Digital Solutions

However, the VeriChip does not interact with its user but is strictly passive. An example of active digestible computing would be the work of Professor Kevin Warwick at the University of Reading. He is one of the leading researchers in Cybernetics and in March 2002, he had a silicon chip implanted to nerve endings in his left arm.


Array connection to median nerve
Exit from skin
Fixed connector
Implant site
Detachable radio transceiver module
Figure 13. Implant Diagram, Cyborg project (available from http://www.rdg.ac.uk/KevinWarwick/html/project_cyborg_2_0.html)

The chip is then connected to radio transceiver on the exterior of the arm, which can then send the signals from the nerves detected by the chip to a computer. These signals will be analysed by the research group in order to learn how nerve signals can be used to control or interact with other devices, perhaps internally or externally to the host body. This could aid those with physical disabilities to control their own limbs or objects in their surrounding environment.

4 Evaluation scopes of the ubiquitous systems

Evaluation scopes of ubiquitous computing systems have become increasingly important as various ubiquitous devices and application software start to be emerged. In addition, the evaluation scopes of the ubiquitous systems would provide the designers with essential system specifications and for the evaluators, in effect, such as Telecommunication Technologies Association (TTA) could derive the new user-centred guidelines such as Macintosh human interface guidelines (Apple computer, 1992) with regards to the ubiquitous systems.

As discussed above, the ubiquitous systems have many aspects to be considered other than the software or application itself. For instance, Davies and Gellersen (2002) examined a number of deployed ubiquitous computing systems, identifying each ubiquitous computing systems' main characteristics and what we have to consider in developing a ubiquitous computing system.

This implies that we cannot directly apply the previous standards or the scope of evaluation of desktop computing environment to the ubiquitous systems. Partially, this comes from the previous assumption of the interactions between the systems and the users are almost one-to-one, however, under the ubiquitous environment, we must consider the one person– to–many computing interaction. For this end, Scholtz and Consolvo (2004a, 2004b) have suggested a framework or an evaluating scope, to some extent, for assessing ubiquitous computing application, proposing a relatively comprehensive structure. Yet, the framework or the scopes can only be applied to the level of ubiquitous computing application, while others such as services and the technical side of ubiquitous computing systems are not clearly incorporated. In this report, we intend to embody the general components of the ubiquitous systems to embrace both the application level and the system level. Subsequently, each of levels is deployed with appropriate characteristics in consideration.

In addition to this, we note "control theory" which is widely used in engineering and mathematics domain, on the grounds that it can easily deal with the behaviour of dynamic systems over output of a system.

Figure 14 is a representation of our proposed framework. A general ubiquitous computing environment described by this framework has five entities and ten functions correspondingly.

Activities are conducted in the ubiquitous system through the users. From a user's perspective, an interaction with ubiquitous systems is an interleaving of activities and evaluation functions. The user's activities trigger receptive events in the system and the users evaluate expressive events from output artefacts in the system.

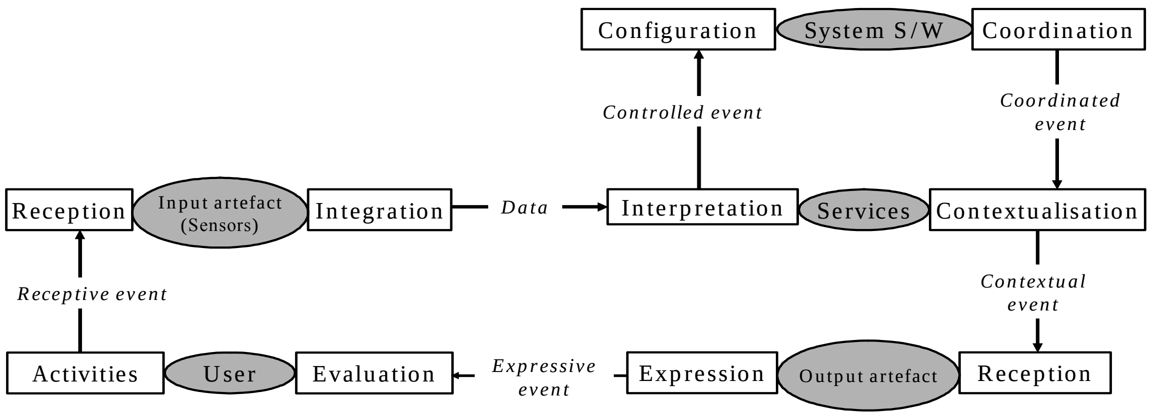


Figure 14. Representation of the control loop in ubiquitous systems

The output artefacts, e.g., large-wall-mounted display, lights, sound or even haptic interfaces, have receptive and expressive functions. The receptive function accepts the formulation and acts upon contextual events triggered from the ubiquitous service. The expressive function represents the state of the system resulting from the contextual event. These processes between the user and the physical devices have been well understood in the previous HCI literature.

From the other entities in the ubiquitous system depicted in Figure 14, we can also notice the reason why the design of the ubiquitous environment is increasingly demanding a systematic approach rather than a common human-computer interaction perspective.

When receptive events are triggered by users' activities, sensors or some physical input artefacts carry out the operations of reception and integration of the user's activities. Examples of some physical input artefacts are switches mounted to a grip, trackballs, mobile phone, gesture and speech recognition or context awareness facilities. The receptive function accepts the user's activities and acts upon it, while the integrated function takes into account the states of the several sensors as well as the input artefacts at the same time, delivering the information to the ubiquitous service.

Once the sensors or the physical input artefacts locate a detectable event, the ubiquitous services act upon it to process the integrated event. Ubiquitous services should contain two functions: interpretation and contextualisation. Most of the ubiquitous computing systems receive a lot of data from their sensors or other physical input artefacts. These data thus need to be interpreted or reprocessed to fit into the context of the user's activities. The sensing networks can perform the basic level of data processing while communicating with other sensing components. The context awareness in the ubiquitous services is of great importance. A very significant part of the context is to know or be able to infer the user's intent. For instance, Global Positioning Systems (GPSs) give us locational context. Some good sensing networks provide environmental context, and clever service design can support some levels of intent inferencing, i.e.,

situational context. These ubiquitous services contribute to the promises of ubiquitous systems.
Finally, to control these individual entities in the ubiquitous system, ubiquitous system software and networkware, e.g., operating systems or middleware, has to co-ordinate all the data processing and commands between the components. Therefore, configuration and co-ordination are considered to be essential part of the system software. Specifically, contextualisation functions in services are co-ordinated by system software so that the services can make intelligent decisions on how to interpret events.
Not only satisfying the quality characteristics described in its system specifications, control theory can also guarantee *stability* in the whole ubiquitous computing process of the control loop. For instance, sensors in the ubiquitous system can be densely deployed in an area to observe user's activities, therefore, the input data from one sensor node may be interfered by the others. It may lead to unexpected interpretations of user's activities, resulting in instability in the system. To avoid this possible instability, in general, the ubiquitous system employs sensor management protocol in "integration" functions in order to introduce the rules related to data aggregation, to exchange data related to the location finding algorithm, and to perform time synchronisation among the sensor nodes (Akyildiz et al., 2002).
In this way, this control loop approach can represent which event triggers what interaction as well as what the interaction results in the ubiquitous systems with the proviso that any interaction be directed.
As evaluator using this framework must decide how to collect particular conceptual measures that instantiation becomes the implication-specific measure (Scholtz and Consolvo, 2004b). For instance, in the desktop environment, if time-critical, efficiency must be optimised; when errors are unacceptable, effectiveness is the most important evaluation scope. However, the unique characteristics of the ubiquitous systems might require more comprehensive measurements. We expect that TTA would assess any ubiquitous applications from our pools of the evaluation scopes in the ubiquitous systems. The following sections thus outline how these five entities with corresponding functions interact among them in order to achieve the primary goals of the ubiquitous system, and in turn delineate the evaluation scopes of the ubiquitous computing systems.

4.1 User

In the context of human-computer interaction, a strong need for human-centred design has always been emphasised. This is equally applicable to the design of ubiquitous systems, even though the ubiquitous systems do not call for the same user attention as the conventional computing does. This difference between the traditional interactive system and the new ubiquitous system leads us to pay attention to the user's functions in terms of user's activities and the evaluation of system feedback.

4.1.1 Activities

In the ubiquitous environment, users carry out activities less likely to be the same as those in a desktop environment. Activities in the ubiquitous environment are complicated, because the users may handle other physical or mental tasks while interacting with pervasive artefacts, which they might use in a variety of situations. To best support the practices of user's activities in the ubiquitous computing environment, important attributes like *predictability*, *accountability*, and *simplicity*, of the activities must be satisfied.

Predictability

In the traditional HCI terms, it has long been recognised that the match between the conceptual model of the user and the system can increase the predictability of the system. This will help the user predict the effects of their activities.

Building upon this, equally, the ubiquitous system should be *predictable* where the user can determine what the effect or impact of a future input/action would be. This is of great importance in the ubiquitous systems where there would be a greater potential for unpredictability, due to the fact that the ubiquitous systems often adapt their behaviour according to the context of use, including such things as the user's location, identification, orientation, proximity to other devices and people. Even if a system adapts its behaviour according to a well-specified algorithm, if the user cannot understand why the system behaves in a particular way in one situation and in a different way in a seemingly similar situation, the user will be confused. In this sense, context-aware systems might have varying scores in predictability, but high efficiency and effectiveness scores in the ubiquitous service (see Section 4.3).

The new concept, *feedforward* (Djajadiningrat et al., 2002), would be very useful, in order to make the users aware of how their system would behave when they act on it. They claimed that the sensory richness and action potential of physical objects can act as carriers of meaning in interaction. In this context,

Understandability or *Justifiability* is strongly related to predictability, and predictability measurements can be indicators of the understandability or justifiability potential of the ubiquitous system.

The most global form of predictability concerns user's ability to assess systems' overall level of competence – the degree to which the system tends in general to perform its tasks successfully (Jameson, 2003). For instance, if the user seriously overestimates system's competence, they may rely on the system excessively; if the user underestimates the system, the user will not derive the potential benefits that the system can provide.

In general, the levels and degrees of predictability that is necessary or desirable in a give case can depend on many factors, including functions that are being served by the ubiquitous service and user's level of skill and experience. The same is true of the choice of the measures that are most appropriate for the achievement of predictability. However, at least, the following questions and measurements should be considered for ubiquitous computing systems.

Evaluation question(s):

- Can users predict or understand what is required as input data and what is provided as output by the ubiquitous systems?
- Is the ubiquitous computing system possible to provide feedback to be justifiable?

    - Conduct user test and interview with questionnaires or observe user behaviour

Metric(s):

- A/B where A= Number of input and output items which user successfully understands and predicts, B= Number of input and output items available from the ubiquitous interface
- A/B where A= Number of input and output items which user successfully justifies their activities, B= Number of input and output items available from the ubiquitous interface

---

Evaluation question(s):

- How consistent are the components of the user interface
    - Observe the behaviour of the user and ask for opinions

Metric(s):

- Average acceptance rate of users for consistency
- 1-(A/B), where A= number of messages or functions which user found unacceptably inconsistent with the user's expectation, B= number pf messages or functions
- N/UOT, where N= number of operations which user found unacceptably inconsistent with the user's expectation, UOT= User operating time (during observation period)

Evaluation question(s):

- Are users hesitating to do some activities in the ubiquitous systems?

Metric(s):

- The number of times that the user pause for a long period with hesitation during observation period

---

Evaluation question(s):

- Can users predict or understand what the vocabulary from the ubiquitous systems mean?

Metric(s):

- 1 –(A/B), where A= number of vocabulary which user unsuccessfully found the meaning of the vocabulary, B= number of vocabulary used in the system

---

<u>Accountability</u>

We are already aware that a system is usable if the user can easily understand, learn, use and be attracted, under the particular context of use. This is of great importance in the ubiquitous systems where there would be a greater complexity for different meanings in the different contexts or situations. This is due to the fact that the ubiquitous systems

often interpret user's intentions in different ways according to the context of use, including such things as the user's location, identification, proximity to other devices and so on.
Accountability refers to the extent to which the user can provide the reason why their activities bring about states of the system, in a particular way in one situation and in a different way in a seemingly similar situation. Although accountability tends to be enhanced by predictability, these properties are not perfectly correlated. For instance, when the user is engaged in more important tasks, the user can predict that a Portable Help Desk (PHD: Smailagic et al., 2000) would deliver relevant information to them; however, the user has no accountability over how this information has been made.
A typical measure for ensuring some degree of accountability is to have the system provide any action with significant consequences to the user for approval; however, this has negative impact on the *unobtrusiveness* in the artefacts, so that it is a complicated design challenge to find ways of making recommendations in an unobtrusive fashion that still makes it easy for the user to notice and follow up on them. A sonified way has been a most likely possibility.
Like predictability, accountability can be achieved on various levels of granularity. For example, specific behaviours of the system should be accountable (system level) and user's behaviour should also be accountable in the system with a meaningful way (user level). In particular, it is important to consider what kinds of situation awareness can be provided by the system to keep the users aware of the outcomes and meanings of their activities. The same is true of the choice of the measurements that are most appropriate for the achievement of accountability, However, at least, the following questions and measurements should be considered.

Evaluation question(s):
- Can users understand how the current states of the system were provided as output by the ubiquitous systems?

Metric(s):
- A/B where A= number of input and output items which user successfully understands and predicts, B= number of input and output items available from the ubiquitous interface

---

Evaluation questions
- Is the ubiquitous computing system possible to ask user's approval in the autonomous situation where there are competing contexts?
  - Conduct user test and interview with questionnaires or observe user behaviour

Metric(s):
- A/B where A= number of approval event, B= number of autonomous situation where there are competing context

---

Evaluation question(s):

- Does the ubiquitous system provide any user support or help facility to make sure what the user is activating in the system?

Metric(s):

- A/B, where A= number of trials which user successfully found system help/support, B= number of trials to access to system help/support

---

<u>Simplicity</u>

In the desktop user-centred design guidelines, the best approach is to make the system easy to use. This general design guideline that simple design is good design applies directly to the disciplines of the ubiquitous system. The challenge that users desire is to solve their problems using the systems to facilitate their work. The more simplified the users' activities are, the more likely that we can build the ubiquitous systems to meet their needs and expectations. Thus, cognitive costs (e.g., memory load) and physical costs (e.g., movement) are most likely to be minimised. The current TTA guideline (U.1.5: VeriTest, 2002), concerns with regards to human memory loads.

In GUI application, a set of menus and toolbars provide access to the functions of the application. In most cases, users perform an action on an object by selecting the object and then selecting which action to apply to it. This issue becomes apparent as soon as designers attempts to create invisible interfaces, in which the GUI disappears into the environment. In such settings the user is not looking at a computer screen, thus conventions cannot be pertinent.

In the HCI domain, *progressive disclosure* is known to be the best way to reduce the complexity of user's activity (Apple computer, 1992). It allows users to see the most common choices while initially hiding more complex choices or additional information. It helps for novice users to learn and includes the features and power that advanced users desire. On the other hand, *preference settings* help the complexity of user's activities be lessened. In order to reduce the complexity of the systems, make decisions about which features to implement as preference based on what the users really need. These two common approaches to avoid complexity of user's activities are also suitable to the ubiquitous systems; however, they may be hampered by the contextual output from the ubiquitous service, in that the responses from the system would be dependent on contextual awareness. In addition, the keyboard shortcut to activate some actions in GUI can be applicable to the ubiquitous systems. Of course this should be adjustable, based on the frequency of use in the ubiquitous systems.

The levels and degrees of simplicity can take many different forms, including how to handle complex operations in the ubiquitous systems or how to provide simple way of performing a task. The same is true of the choice of the measurements that are most appropriate for the achievement of simplicity. However, at least, the following questions and measurements should be considered.

Evaluation question(s):

- How long does the user take to use activities correctly?

    - Conduct user test and observe user behaviour

Metric(s):

- Mean time taken to use a function correctly
    - Operation report, user monitoring record

Evaluation question(s):

- How long does the user take to learn how to perform the specified task efficiently?
    - Observe user behaviour from when they start to learn until they begin to operate efficiently

Metric(s):

- Mean user operation time until user achieved to perform the specified task within a short time

---

Evaluation question(s):

- How complex is it to specify or activate a given task?

Metric(s):

- Mean number of steps to activities

Evaluation question(s):

- Can user easily reduce operation procedures for his/her convenience?
    - How to handle complex operations (e.g., multiple objects, actions, and more abstract functions that are difficult to represent graphically)

Metric(s):

- A/B, where A=number of reduced operation procedures after customising operation, B= number of operation procedures before customising operation

---

4.1.2 Evaluation

In a ubiquitous environment, another function belongs to the user category is to evaluate the states of the system, in general, based on the different outputs from devices like large-wall-mounted display, lights, sounds sensors. In the HCI literature, evaluation is strongly associated with the subsequent goal reorganisation (Ryu and Monk, 2004). That is, based on the evaluation of the states of the system in conjunction with the current goal state, the users perform the next activities to best match to the overall goal. We believe that when a ubiquitous system becomes more sophisticated, the low-level interaction issues will surely be considered. To address this issue, *reversibility* and *acknowledgement* are taken into consideration.

Reversibility

The traditional HCI evaluation has emphasised that the user's activities should be *reversible* to be able to restore to pre-existing states of the system. The recovery guideline (U.1.4: VeriTest, 2002) states that when an error occurs, tell the user what the error is and how to correct it; let the user control the whole system. This guideline is also pertinent to, but more important in the ubiquitous system, with the proviso that the ubiquitous system has many interactions engaged at a time, captured by the sensor network and interpreted by the system. Subsequently, there is very likely to have unintended outputs on the system. This not only prevents mistakes in the first place, but also informs users about mistakes that have already occurred so that they can correct them.

In order to correct mistakes in the ubiquitous systems, they have to be visible in time to take action before it is too late; perhaps during or immediately after a system response, and sometimes even before. With GUI, it can be dealt with in a concrete manner, but in the ubiquitous system without GUI, ambiguity would be a serious problem. Both the action and its object need to be represented in a manner such that they can both be identified and specified as targets for undo. For instance, Digital Voices (Lopes and Aguiar, 2001) applications are appealing because constant feedback is provided, which should allow the user to cancel an error in progress. However, it is not clear how users could differentiate system communications that contain erroneous content from correct ones.

Conventionally, undo function with GUI is the most preferred way to provide reversibility; however this convention may not be communicated with the ubiquitous users, which leads to a designer challenge to provide a dedicated undo mechanism in the ubiquitous systems.

This reversibility issue can be relatively easily measured, and the measurement would provide a better idea for the design of a ubiquitous system. At least, the following questions and measurement should be taken for consideration.

Evaluation question(s):

- Can user easily correct errors on tasks? (after executing an activity)
    - Conduct user test and observe user behaviour

Metric(s):

- Tc-Ts , where Tc= time of completing correction of specified type of errors from performed tasks, and Ts= time of starting correction of specified type of errors from performed tasks
- A/UOT where A= number of times that the user succeeds to cancel their error operation, UOT= user operating time during observation period

---

Evaluation question(s):

- How frequently does the user successfully correct input errors? Or how frequently does the user correctly undo errors? (before executing an activity)
    - Observe the behaviour of the user who is operating the ubiquitous system
- Is there any way to restore the previous state without further difficulty?
    - How to control or cancel system action in progress
    - How to disambiguate what to undo in time
    - How to intervene when users makes obvious error

Measure(s):

- Degree of enabling undo services
- Degree of supports to users in correcting errors

Metric(s):

- A/B, where A= Number of instances where the input data were successfully modified or changed before being elaborated, B= number of instances where user tried to modify or to change the input data during observed user operating time
- A/B, where A= number of input errors which the user successfully corrects, and B= number of attempts to correct input errors

---

<u>*Acknowledgement*</u>

To see whether the ubiquitous system has received and processed user's request, it should provide appropriate acknowledgement for the user to recognise what they have done. This enables the users to understand the actions and states of the system and allows them to avoid errors and plan for their next actions. The current guideline, for instance, informative feedback (U.1.3: VeriTest, 2002) keeps users aware of whether the system is processing or the system has completed a request. Yet, this challenge would be cumbersome, by definition where the ubiquitous system is everywhere, embedded in mundane objects, so the goal of making state perceivable seems very daunting. Bellotti et al., (2002) maintained that without a GUI equivalent it is very difficult to know how the system is responding to their ubiquitous actions, in particular, augmented objects, gestural UIs and sonified I/O do not presuppose any mechanism to display state information in a manner that is consistent with the mode of input. Acknowledgements from the ubiquitous system should be very instantaneous. Particularly, when one-to-many human-computer interactions are triggered in an ubiquitous system, slow feedback will lead to lots of confusions at the user's end.
The levels and degrees of acknowledgement can take many different forms. The levels and degrees of acknowledgement can take many different forms. For instance, the enforced acknowledgement costs more attention to users, which is less likely to be expected in the ubiquitous computing system. However, if it is inevitable, it must be timely and ask less interference. The same is true of the choice of the measurements that are most appropriate for the achievement of acknowledgement. However, at least, the following questions and measurements should be taken for consideration.

Evaluation question(s):

- Can user easily understand messages from the ubiquitous systems?
- Is the information useful and relevant to the users?
- Is the system designed to convey "just enough" information?
- Can user easily notice the state of the system?
    - The transition from one state to another should be easily perceptible

Metric(s):

- A/UOT, where A= number of times that the user pauses for a long period or successively and repeatedly fails at the same operation, because of the lack of message comprehension, UOT= user operating time (observation period)

Evaluation question(s):

- Is there any message which caused the user a delay in understanding how to do the subsequent action(s)?
    - Observe the behaviour of the user who is operating the ubiquitous systems

Metric(s):

- 1-(A/B), where A= number of messages that the user found unacceptable according to his/her expectation, B= total number of messages

---

Evaluation question(s):

- What is the average waiting time the user experiences after issuing a request until the request is completed within a specified system load in terms of concurrent tasks and system utilisation (responsiveness)?

Metric(s):

- Tr = (Time of gaining the result) – (Time of command entry finished)

4.2 Input/Output artefacts

Devices in the ubiquitous systems supply power for computation, communication, and perception in much the same way that batteries and wall outlets supply power for electrical appliances. Both mobile and stationary devices are universal communication and computation appliances. They are also anonymous: they do not store configurations that are customised to a particular user.

Collections of embedded devices will create intelligent spaces inside offices, buildings, homes, and vehicles. They provide large amounts of embedded computation, as well as interfaces to camera and microphone arrays, large area displays, and other devices. Users communicate naturally in the spaces created using speech and vision, without being aware of any particular point of interaction.

Handheld devices provide mobile access points for users both within and without the intelligent spaces. They accept speech and visual input, and they can reconfigure themselves to support multiple communication protocols or to perform a wide variety of useful functions (e.g., to serve as cellular phones, beepers, radios, televisions, geographical positioning systems, cameras, or personal digital assistants).

The severe resource restrictions of computer-augmented everyday artefacts imply substantial problems for the design of applications in smart environments. Some of these problems can be overcome by exploiting the resources, I/O interfaces, and computing capabilities of nearby mobile devices in an ad hoc fashion.

Speech and vision, rather than keyboards and mice, provide the main modes of interaction in the ubiquitous systems. Multimodal integration increases the effectiveness of these perceptual technologies, for example, by using vision to augment speech understanding through recognising facial expressions, lip movement, and gaze.

As Weiser (1991) depicted in his paper, the physical devices that can make users connect to the ubiquitous computing world are the most crucial factor of the success of the ubiquitous systems. However, the ubiquitous system, where input is sensed by means of other than keys, or mouse, and output is expressed on other than a desktop monitors, has little understood. At least, we believe that sensing nodes, PDA, GPS would be the most plausible devices, and then have developed individual design guidelines, respectively. Lacking these well-established evaluation scopes, evaluators of ubiquitous system would have constantly been found inconsistent with their assessments.

The sensor network is the most preferred one for its extensive use as the ubiquitous input artefact. On the other hand, some wearable computing researchers are developing wearable I/O devices for their ubiquitous environment. In terms of the functions they are performing in a control loop, the physical I/O artefacts can keep users connect to the ubiquitous computing world. In this sense, different types of I/O artefacts are jointly considered in this section.

Many researchers have suggested some design implications of the physical artefacts for the ubiquitous system. Yet, none of these studies has a fully integrated view of all the factors driving the design of sensors and physical I/O devices. These factors are important because they serve as a guideline to design a ubiquitous service protocol or an algorithm for sensing and output. The following section outlines the roles of sensors and physical I/O artefacts into three categories: reception, integration, and expression.

4.2.1 Reception

The issue we raised here is so fundamental that it is often taken for granted in interface design: what mechanism does the user employ to address the system? (Bellotti et al., 2002).

In the desktop computing environment, the system concept is so clear that designers know that if the user intends to interact with the system, he or she should use devices like the keyboard or mouse. There is little possibility for error, users seldom accidentally touch other input devices. Such assumptions, however, are invalid when more ambient modes of input, such as gestures and movements, are used. Unobtrusively attached tags and sensors make it harder for users to distinguish objects that the system is attending to from ones that the system is ignoring. Without visible affordances users can unintentionally interact or fail to interaction. In such settings, the following aspects must be considered: quality of reception, and mobility.

Quality of reception

The fundamental objectives for the ubiquitous input devices, such as sensor networks or gestural input, are *reliability, accuracy, flexibility, fault tolerance*, efficiency, effectiveness, dynamic sensing scheduling, resource optimisation and ease of deployment. The first four would be considered as the characteristics of quality of reception, the others are discussed further in Section 4.3 (Service) and 4.4 (System software and Network).

Firstly, one problem for the sensing input approach is a risk of failure to communicate with the system if a sensor fails (Input reliability). In addition, targets should be well spaced and use limited sensing ranges or durational thresholds because getting too close can lead to accidental addresses. Individual sensors are unreliable, particularly in harsh and unpredictable environments. Addressing sensor reliability can reduce the level of redundancy required for a network to operate with the same level of reliability. A reversal problem is avoiding unintended communications with devices that the user does not want to interact with. Accidentally addressing a system could be more than annoying; it could be a serious hazard. For instance, a voice activated car phone triggered accidentally could compete for a driver's attention with serious consequences. Robust sensor design, integrated with high levels of fault tolerance and network reliability enable the deployment of sensor networks in dangerous and hostile environments, allowing access to information previously unattainable from such close proximity (device resolution). Reliability can be achieved on various levels of granularity. The same is true of the choice of the measurements that are most appropriate for the achievement of reliability. However, at least, the following questions and measurement can be taken for consideration.

Evaluation question(s):

- How often does the user succeed to exchange data between target and other component? Or can user usually succeed in exchanging data?

Metric(s):

- A/B, where A=number of cases in which user succeeded to exchange data with other systems, B= number of cases in which user attempted to exchange data

---

Second, physical I/O artefacts are predominantly data-centric. Given the similarity in the data obtained by sensors in a dense cluster, aggregation of the data is performed locally. That is, a summary or analysis of the local data is prepared by an aggregator node within the cluster, thus reducing the communication bandwidth requirements. Aggregation of data increases the level of accuracy and incorporates data redundancy to compensate node failures. The utilization of a larger number and variety of sensor nodes provides potential for greater accuracy in the information gathered as compared to that obtained from a single sensor. The ability to effectively increase sensing resolution without necessarily increasing network traffic will increase the reliability of the information for the end user application.

There may be many forms of accuracy. The same would be true of the choice of the measurements that are most appropriate for the achievement of accuracy, however, at least, the following questions and measurement can be taken for consideration.

Evaluation question(s):

- Are differences between the actual and reasonable expected results acceptable?
  - Do input vs. output test causes and compare the output to reasonable expected results
- How often do the physical artefacts produce inaccurate results?
  - Record the number of inaccurate computing or sensing based on specifications

- What is the absolute limit of transmissions required to fulfil a function?
    - Sensing resolution (distance of sensing)
- What is the degree of synchronisation between different media over a set period of time?
- Is the ubiquitous system capable of performing tasks within expected transmission capacity?
- What ranges does the physical artefact accurately receive the data?
    - Interference and integration level between the physical artefacts

Metric(s):

- A/T, where A=number of cases encountered by the users with a difference against to reasonable expected results beyond allowable, T= operation time
- Count the number of cases encountered by the users with an unacceptable difference from reasonable expected results
- A/T, where A=number of inaccurate computations encountered by users, T = operation time
- Count the number of events where the physical artefact did not receive the data from the system, according to the system specification of the communication coverage

---

Third, the ubiquitous system imposes, basically, many demands on the physical artefacts to display some information to the user or detect users' activities to process. Some of the physical artefacts are dedicated in sensing users' activities; others would be more flexibly used in the environment. For instance, it can be seen that some PDAs are very flexible artefacts in the ubiquitous system in order to communicate with their GPS module to locate their position, i.e., sensing, and then display the current location on the readout of the PDA, i.e., expression. By contrast, sensing artefacts fixed on the environment are only dedicated to detect users' activities. As flexibility is required in the physical artefacts, the following concerns are dealing with how flexibly the artefacts can be used in the ubiquitous system.

Evaluation question(s):

- Are the devices wearable or portable?
    - Can the user sustain the weights of the devices?
    - Is the level of power consumption within the range of the system specification?
    - Do the physical artefacts conform to the international standards such as radiation issue? (Health/Safety issue)
- Can they easily interact with other artefacts from the systems?
- How much screen 'real-estate' space is available for displaying information on the hand-held device?
    - Can normal web pages be displayed? And what is the resolution?
    - Can graphics be displayed?
    - Documents are readable?

Metric(s):

- User ability to change input devices as improvements
- User ability to sustain the weights of the devices
- Number of tasks user can accomplish that were not originally envisioned
- Number of hours for the portable artefacts to survive without extra power supply

---

Finally, fault tolerance of the physical I/O devices should also be considered. Of course this issue is dealt with in Section 4.4 (network), the following issues are more relevant to be discussed in this section, focusing on the I/O devices.

Evaluation question(s):
- How many hardware (I/O artefacts) failures were detected during defined trial period?
- What kinds of mechanisms were considered to avoid critical and serious failures?
- How often does the user encounter problems in I/O device related operations?

Metric(s):
- A1/A2, where A1= number of detected failures, A2= number of performed test cases
- A/B, where A= number of avoided critical and serious failure occurrences against test cases of fault pattern, B= Number of executed test cases of fault pattern during testing.

Mobility

Most contemporary mobile devices feature wireless connectivity. It implies that a device can be easily transported to a location where the user wants to interact with. Only those devices that support mobile interactions are fully mobile; devices that can be moved to a different location, but require the user to remain stationary during the interaction are no more than transportable. On the usability side, we see some critical differences between stationary interactions, where user movement is restrained and mobile interactions, where various degrees of body movement are allowed, particularly, when the user is walking.

In the ubiquitous computing environment, multiple sensor networks may be connected through sink nodes, along with existing wired networks. The clustering of networks enables each individual network to focus on specific areas or events and share only relevant information with other networks enhancing the overall knowledge base through distributed sensing and information processing.

Apart from the sensor network, the physical artefacts should be able to support mobility. The capability of the ubiquitous system to enable the user to easily locate, identify and access it seamlessly as a whole. A distributed wireless network incorporating sparse network properties will enable the sensor network to span a greater geographical area without adverse impact on the overall network cost. At least, the following questions and measurement can be taken for consideration when checking the mobility of physical artefacts.

Evaluation question(s):

- Is the device easily mobile?
- Is the devise transparently passed between sink nodes in the system?

Metric(s):

- Device handover rate = A / B, where A = Number of successful device handovers, B = Number of attempts to handover

---

4.2.2 Integration

Ubiquitous systems must be connected dynamically while changing configurations of self-identifying mobile and stationary devices to form collaborative regions. In this sense, the integration of data received by each network increases the level of accuracy and incorporates data redundancy to compensate node failures. This is highly related to the network which is discussed in Section 4.4.

In most of the literature we have seen, the partitioning and merging of networks has not been addressed in detail. One potential area of research here is in the service discovery, if two networks are to merge together there will presumably be a reason to do so, for example one network may offer printing facilities to another network, issues of service discovery are important here. Other potential areas here include methods for detecting partitioning and merging of networks.

<u>Ubiquitous connectivity and Integrity</u>

Ubiquitous systems investigate different forms of wireless communication to suit the needs of ad-hoc communication in ubiquitous computing environments. Therefore, short-range wireless such as Bluetooth and wide-area wireless such as GPS or wired communication are seamless integrated.

In general, the levels and degrees of integrity that is necessary or desirable in a given case can depend on many factors. The same is true of the choice of the measures that are most appropriate for the achievement of ubiquitous connectivity. However, at least, the following questions and measurement can be taken for consideration.

Evaluation question(s):

- Are the physical artefacts seamlessly connected?
- How wide range does the ubiquitous computing system take coverage?
- Does the ubiquitous computing system consider the accuracy of each network equipment, according to the system specification?

Metric(s):

- A/T, where A= number of cases encountered by the users with the disconnection in the system beyond allowable

- Count the number of events where the physical artefact did not receive the data from the system, considering the system specification on communication coverage

---

4.2.3 Expression

According to our control loop framework, the output artefacts have expressive functions, which represents the state of the system resulting from contextual events. Evaluators of user interfaces for standard applications, devices, and systems rarely have to worry about this expressive function, because these are almost guided by interface design guidelines such as Macintosh human interface guidelines (Apple computer, 1992). By sticking to the Graphic User Interface (GUI) using standardised toolkits, and by copying design ideas from existing solutions, evaluators has assessed expressive functions from pre-existing techniques, without pondering new interaction issues. While the ubiquitous systems are employing the basic components of standard user interfaces, we believe that the design guidelines (U.2: VeriTest, 2002) would also be applicable, but must consider different environmental factors (e.g., different I/O artefacts) between the desktop environment and the ubiquitous systems.
In this report, we present some issues anew for the expressive functions addressing the resulting design challenges inherent in the ubiquitous systems.

Transparency

Transparency refers to the user can understand system actions and has a clear picture of how the system works. We already maintained that user's activities must be seen transparent or predictable in the ubiquitous system. This leads the user to understand how their activities would make effects on the system. It is also strongly related to the evaluation function of the user. That is, physical artefacts must provide information about activities which can be performed, as well as about the states of the system unambiguously and clearly. This enables the user to be able to predict the consequences of their planned actions, which is strongly linked to predictability in activities.
We note that the consistency guideline (U.1.1: VeriTest, 2002), which refers to consistent visual appearance and consistent response to user inputs are required throughout the user interface, helps transparency by consistently meeting user's expectation. Yet, it may be hampered by the contextual output from the ubiquitous service, in that the responses from the system would be dependent on contextual awareness.
In general, the levels and degrees of transparency that is necessary or desirable in a give case can depend on many factors. The same is true of the choice of the measures that are most appropriate for the achievement of transparency. However, at least, the following questions and measurement can be taken for consideration.

Evaluation question(s):

- Does the output artefact add minimal cognitive load?
    - Cognitive load may be higher when users must remember what states or

changes in the output mean. Should be intuitive.

Metric(s):

- Effectiveness comparisons on different sets of I/O devices
- Degree of ambiguity of the application (including commands, dialogues etc)
- Using multi-modal interactions?

Unobtrusiveness

The computational basis that supports user's activities will become increasingly transparent. The actual execution of important tasks will migrate from the centre to the periphery of attention (Crook and Barrowcliff, 2001). We will use the term obtrusiveness to refer to the extent to which the system places demands on the user's attention that reduce user's ability to concentrate on user's primary tasks (Jameson, 2003). They can also do so without having to watch the system, for instance, they might be attending to other matters, thus the audio channel can be an appropriate alternative to visual displays. A ubiquitous system is to transform users into first class entities, who no longer need to exert much of their attention to computing machinery. Therefore, even the system should be transparent to a certain level, blending into the background without distracting users too much. In this context, wearable computing environment provide high profile of unobtrusiveness.

This issue focuses only on addressing the system. In addition to this, users must determine whether and when the system is attending to them. Somehow the system must provide cues about its attention; analogous to an audience sending signals of his/her attention to a human speaker. In GUI solution, such as flashing cursors and watch icons are part of the established mechanisms for communicating whether a system is accepting or responding to input. However, there are inherent problems with the ubiquitous systems. Unobtrusively attached tags and sensors make it harder for users to distinguish objects that the system is attending to from the ones that the system is ignoring. Without visible affordances users can unintentionally interact or fail to interaction.

Unobtrusiveness is also strongly related to *invisibility* in the ubiquitous system, which makes inferences about the user's activities, goal, emotional state, and social situation and attempt to act on behalf of user. If the system has sensed and interpreted the context correctly, this initiative can result in time savings and a reduction in user workload, in turn, paying less attention to the physical artefacts in the ubiquitous system. In particular, the feature of unobtrusiveness can exist for the user without their computer being fragmented into specialised tools.

In general, the levels and degrees of unobtrusiveness that is necessary or desirable in a give case can depend on many factors. The same is true of the choice of the measures that are most appropriate for the achievement of unobtrusiveness. As the goal of ubiquitous computing is to have the computer disappear, one possible metric could be the amount of distraction necessary for the user to complete a task. Smailagic et. al, (2001) have proposed a distraction matrix in which they classify the time needed for a distraction and look at the categories of information, communication, and creation with respect to these distractions. However, at least, the following questions and measurement can be taken for consideration.

Evaluation question(s):

- How much attention does the user have to pay in order to command their activities?
- Is the display unobtrusive and remain so without requiring user's attention?
  - User should be able to easily monitor the display
- Does one notice a display because of a change in the data it is representing and not because its design clashes with its environment?

Metric(s):

- A/UOT where A= number of times that the user staring at the input or output artefacts to trigger their next activities, UOT= user operating time during observation period

---

Awareness support

Even when the system provides appropriate feedback, the feedback should also be presented in a sufficiently intrusive way to take user's attention in the forms of outputs. *Awareness* is a more complex issue in the ubiquitous systems design, in that users may handle other physical or mental tasks while interacting with pervasive devices, which they might use in a variety of situations. In particular, co-operative work in the ubiquitous system also requires awareness of the actions of other users, the consequences of these actions, and the effect of one's own actions on other people. System behaviour must be sufficiently intrusive to support such awareness amongst users. Designers must carefully investigate effects on user focus to seamlessly blend applications into the environment or to provide peripheral awareness (Scholtz and Consolvo, 2004b). In this context, ambient displays must score well in the areas of both awareness support and unobtrusiveness.

With respect to these challenges, if a state change is a part of the function of a system, then these issues must somehow be explicitly addressed. They proposed ongoing projection of graphical or audio information into the space of action, however, this is presently done at the cost of restricting users to work within the projected area.

The levels and degrees of awareness support that is necessary or desirable in a give case can depend on many factors. The same is true of the choice of the measures that are most appropriate for the achievement of awareness support. However, at least, the following questions and measurement can be taken for consideration.

Evaluation question(s):

- Does user ever fail to watch when encountering important messages?
- Does the system provide clear and unambiguous information concerning user's activities?

- Is system behaviour sufficiently intrusive to support such awareness amongst users?
    - How to make system state perceivable and persistent
    - How to provide distinctive feedback on results and state
    - How to embody appropriate feedback, so that the user can be aware of the system's attention
    - How to direct feedback to zone of user attention

Metric(s):

- Ease of co-ordination with others in multi-user situation
- Number of collisions with activities of others
- Number of events not noticed by a user in acceptable times
- Number of display/actions users need to accomplish an interaction or to check on the progress of an interaction
- Workload imposed on the user attributable to focus
- Number of times a user must change focus due to technology

Controllability

On the one hand, we want to avoid the complexity, on the other, the users want o have control of the devices with which they are interacting. In GUI application, people use the devices, such as a keyboard or mouse. There is little possibility for error. However, the capability of the ubiquitous system to enable the user to operate and control it without many difficulties is crucial. This will give the user the locus of control and this enables them to do their activities as intended. Accidentally addressing action will follow. In the context of controllability, the sensor network might have low scores, but personal devices such as PDA must have high scores.

This is strongly related to the flexibility that refers to the user should be able to choose appropriate devices available at the right time and in the right place. It will enable the users to perform their activities in the most appropriate and efficient way, and encourages the development of specific user strategies. Flexible devices can support user modification with changing needs.

Aid orientations and navigation help is also an important factor to be considered (U.1.6: VeriTest, 2002), which provide orientation aids and instructions to help users maintain a sense of where they are in the system, what they can do, and how they can get out.

The levels and degrees of controllability that is necessary or desirable in a give case can depend on many factors. The same is true of the choice of the measures that are most appropriate for the achievement of controllability. However, at least, the following questions and measurement can be taken for consideration.

Evaluation question(s):

- Can user operate the ubiquitous system long enough without human error?
    - Observe the behaviour of the user who is operating the ubiquitous systems

- Does the output artefact make it easy and quick for users to find out more detailed information, if the system offers multi-levelled information?

Metric(s):
- A/B, where A= Number of instances where the input data were successfully modified or changed before being elaborated, B= number of instances where user tried to modify or to change the input data during observed user operating time

---

Evaluation question(s):
- Can user easily recover his/her input?
    - Observe the behaviour of the user who is operating the ubiquitous system

Metric(s):
- T/N, where T= operation time period during observation, N= number of occurrence of user's human error operation

---

Evaluation question(s):
- How much control does the user have over the placement of service?
- What forms of data input are supported by the ubiquitous devices?
    - How to identify and select a possible object for action?
    - How to identify and select an action, and bind it to the object(s)?
    - Is it text-based or graphical?
    - How easy is it to (i) compose tasks, (ii) initiate the execution of tasks, (iii) delete tasks, (iv) correct the specification of tasks?
- Is some information deliberately submitted from input devices? Or how can user easily terminate activities that were already started?
    - How to avoid unwanted selection

Metric(s):
- Effectiveness of interactions provided for user control

4.3 Services

The goal of ubiquitous computing is the production of devices and networks that are so commonplace and natural to use that they become almost invisible. In an ubiquitous environment, there will be hundreds of tiny computers in an office or home, each doing its own specialised task. The devices being ubiquitous means that these specialised elements of hardware and software connected by wires, radio waves and infrared (wireless), will be so ubiquitous that no one will notice their presence. This vision that makes machines fit into the human environment, instead of forcing humans to enter theirs, will make using a computer as refreshing as taking a walk in the woods (Weiser 1991). Most importantly, ubiquitous computers will help overcome the problem of information overload through a distributed architecture.

The ubiquitous computing service refers to the applications that perform computation or action on behalf of users in a ubiquitous computing environment. In detail, a typical definition of a service, from Hawick, et al. (1999), would include all those non-trivial actions upon data that are manipulated by the ubiquitous computing system, such as data discovery, data input, data movement, data modification, the creation of data and the output of data. Services can be run on behalf of the system as well as the user. Services are usually well-defined in their functionality as well as their inputs and outputs.

Services can usually be chained together to form a complicated task– often represented as a graph of services. The idea is that graphical models can be used to structure the paths along which knowledge can be transferred or propagated as a network. Graphs connect those operations, or services, which are interdependent, ensuring information is passed from the producer to the correct consumer of the data. The system should also provide for the correct translation between data types and representations of the outputs generated by one service that are to be used as the inputs to another. Services, themselves, are a distillation of, at the most basic, the primitive actions of the system that may need to be re-issued, re-produced, or *composed* into more complex sets of operations. Thus, the concept of a graph is itself a self-recursive definition.

The ubiquitous computing service not only provides configuration of integrated event (or data from the sensors or physical devices) but also contextualisation with regard to the information that the ubiquitous computing system has to provide. There are several quality characteristics crucially important for the successful deployment of services in the ubiquitous computing environment: Functionality, service quality, contextualisation (or smartness) and security. These are the key characteristics to make ubiquitous computing possible. Other quality characteristics that should also be considered are outlined by the International Standard Organizations (particularly ISO9126) as Reliability, Usability, Efficiency, Maintainability and Portability.

4.3.1 Functionality

Successfully implemented ubiquitous services performs a variety of tasks without the users' awareness. For example, ubiquitous systems can automatically collect and record data on our daily usage patterns of household TV and toilet to give us a diagnosis report of our health. In order to do so, a ubiquitous environment needs to have hundreds of computers serving us, with each computer designed to provide just a few functions very well.

Within a ubiquitous computing system, a general purpose processor executes the commands at the core. However, from the outside, a ubiquitous computing system presents an obvious and natural capability of performing a specific function (Weiser, 1998).

Functionality is a term used to describe whether the ubiquitous computing service has the features necessary to support the requirements. The general purpose of functionality testing is to verify if the product performs as expected and documented. Developers creating a new product start from a functional specification, which describes the product's capabilities and limitations. Software functionality test engineers utilise this specification as a guideline for expected product response. Tasks are exercised to test specific features or functions, and the results of the tasks are verified to be in compliance with the expected response.

Compliance

First, functionality compliance of ubiquitous computing services refers to the ability to make the ubiquitous computing service adhere to application related standards or conventions or regulations in laws and similar prescriptions. The following questions and measurements should be considered for evaluation purpose.

Evaluation question(s):

- How compliant is the functionality of the product to applicable regulations, standards and conventions?

Metric(s):

- Functional compliance ratio X = 1 –A/B, where A = No. of functionality compliance items specified that have not been implemented, and B = Total No. of functionality compliance items specified

---

Evaluation question(s):

- Do all primary functions perform as expected?
- Do contributing functions behave without system hang, system crash, or instability of application?

Metric(s):

- Success rate = the percentage of test cases that passed at the last execution = No. of failed test cases / Total No. of test cases
- Defect Rate = No. of defects / No. of unique test cases
- Traceability matrix mapping from system requirements to test cases (Reference to I/O artefacts, possible overlapping).

Suitability

Second, the suitability metrics measures attributes of the ubiquitous computing service that bear on the presence and appropriateness of a set of functions for specified tasks, for example, the occurrence of an unsatisfying function or occurrence of an unsatisfying operation. The following questions and measurements should be considered for evaluation purpose.

Evaluation question(s):

- How adequate are the evaluated functions?

Metric(s):

- Function Adequacy X = A / B, where A = No. of functions in which problems are detected, and B = Total No. of functions

Evaluation question(s):

- How complete is the implementation according to requirement specifications?

Metric(s):

- Functional Implementation completeness = A / B, where A = No. of missing functions detected, B = No. of functions required in the specification

---

Accuracy

Third, functionality accuracy of ubiquitous computing services refers to its ability to provide right or agreed results or effects to users. For example, if the accuracy of a number of sensors was low, the network has no alternative but to look at the combination of them to get a better idea of what is happening. In a ubiquitous environment, as the sensors come and go or gain and lose accuracy, a Jini architecture could communicate this information. The following questions and measurements should be considered for evaluation purpose.

Evaluation question(s):

- Are the differences between the actual and reasonably expected results acceptable?

Metric(s):

- Accuracy to expectation Ratio = A / T, where A = No. of cases encountered by the users with a difference against the reasonably expected results beyond allowable, and T = Operation time

---

Evaluation question(s):

- How often do the end users encounter inaccurate results?

Metric(s):

- Computational accuracy Ratio = A / T, where A = No. of inaccurate computations encountered by the users, and T = Operation time

---

Evaluation question(s):

- How often do the end users encounter results with inadequate precision?

Metric(s):

- Precision X = A / T where A = No. of results encountered by the users with levels of precision different from required, T = Operation time

---

4.3.2 Service Quality

For varying application computation and communication demands and for a varying quality offered by the ubiquitous computing environments, the ubiquitous system has to provide multiple grades of service quality and maintain a number of agreed-upon or negotiated service qualities.
For example, business users might require guaranteed service quality to operate their business, but normal web surfers may only require best-effort service. Technically, quality of services (QoSs) can be guaranteed through bandwidth, ratio frequency and response time requirements. In terms of evaluation objectives, several characteristics of service quality should be considered. These characteristics include: Efficiency, resource availability, reliability, maturity, fault tolerance, maintainability and adaptability.

Efficiency

Efficiency refers to the capability of the ubiquitous computing service to provide appropriate performance, relative to the amount of resources used, under stated conditions. An efficiency metrics should be able to measure attributes as the time consumption and resource utilisation behaviour of ubiquitous computing system including software during testing or operations.
Efficiency can also be represented by the rate or speed at which a ubiquitous computing service enables a user to accurately and successfully complete a task. While faster response time is usually better, consistent response time is also important. There are several important measures of efficiency: Time behaviour, capacity, resource utilisation and optimisation.
Time behaviour refers to the capability of the ubiquitous computing service to provide appropriate response and processing time and throughput rates when performing its function, under stated conditions. Capacity is defined as the capability of the ubiquitous computing service to receive and to process an amount of appropriate information without bottlenecking. Resource Utilisation is defined as the ability to support the availability of required resources such as processing capability and/or storage capacity. Optimisation refers to the search for the global optimum, or best overall compromise within a (typically) multi-valued system. Where interactions occur many optima are typically present and this process has no analytical solution, generally requiring adaptive solutions. The following questions and measurements should be considered for evaluation purpose.

Evaluation question(s):

- What is the time taken to complete a task?
- How long does it take before the system response to a specific operation?

Metric(s):

- Response Time T = (time of gaining the result) – (time of command entry finished)

Evaluation question(s):

- How many tasks can be successfully performed over a given period of time?

Metric(s):

- System Throughput X = A /T, where A = No. of completed tasks and B = observation time period

---

Evaluation question(s):

- What is the absolute limit on memory required in fulfilling a function?

Metric(s):

- Maximum memory Utilisation X = Amax /Rmax, where Amax = MAX(Ai), (for i=1 to N), Rmax = required maximum No. of memory related error messages from 1st to i-th evaluation, and N = No. of evaluations

---

Evaluation question(s):

- What is the average number of memory related error messages and failure over a specified length of time and a specified load on the system?

Metric(s):

- Mean Occurrence of memory error X = Amean /Rmean, where Amean = ∑(Ai)/N, Rmean = required mean No. of memory related error messages, Ai = No. of memory related error messages for i-th evaluation, and N = No. of evaluations

---

Evaluation question(s):

- How many memory errors were experienced over a set period of time and specified resource utilisation?

Metric(s):

- Ratio of Memory Error/time X = A /T, where A = No. of warning messages or system failures, and T = User operating time during user observation

---

Service resource availability

Availability is one of the most important characteristics of ubiquitous computing service. If the device is not available, do the other issues matter? Service resource availability is a measure of the probability that a service is running as required. Generally the equation is: Time Running / Time Measured (time running divided by time measured). Thus, if you measured something for 20 minutes and it was only up for 19 of them, you'd have 95% availability. The following questions and measurements should be considered for evaluation purpose.

Evaluation question(s):

- What is the availability of the system?

Metric(s):

- Availability A = MTTF/(MTTF +MTTR), where MTTF is the mean time to failure. MTTR is the mean time to repair.

Evaluation question(s):

- What is the Service Discovery Rate?

Metric(s):

- Service discovery rate = A / B, where A = No. of successful discoveries

B = No. of total discoveries
Service discovery rate is referring to the rate of automatically finding a service for a request based on best match.

<u>*Reliability*</u>

Reliability is referring to the probability of failure-free operation of a computer program for a specified time in a specified environment. Our goal is to define metrics that can quantify the progress of the product toward a specified quality objective.
The numerical value of reliability is expressed as a probability from 0 to 1 and is also sometimes known as the probability of mission success. Reliability is the probability, assuming the system was operating at time zero, and it continues to operate until time t, classifying it into maturity, fault tolerance, and fault recovery. The following questions and measurements should be considered for evaluation purpose.

Evaluation question(s):

- How to measure system reliability?

Metric(s):

- Reliability $R(u) = \exp(-u\lambda(x)) = \exp(-u/MTTF)$. Here u is the projected execution time in the future, x is a variable of integration, and $\lambda(x)$ is the failure rate.
- $MTTF = 1/\lambda(t)$, The MTTF is the mean time to failure of the software (i.e., the average active time until a failure occurrence).

<u>*Maturity*</u>

Maturity metrics are internal measures defining a set of attributes for assessing the maturity of the software. These attributes of the ubiquitous computing service bear on the frequency of failure by faults. The following questions and measurements should be considered for evaluation purpose.

Evaluation question(s):

- How many faults were detected in this product?

Metric(s):

- Fault Detection Rate = A/B, where A = absolute number of faults detected in this product, and B = number of estimated faults to be detected (using past history or modelling techniques)

Evaluation question(s):

- How many faults have been corrected?

Metric(s):

- Count No. of corrected faults

Evaluation question(s):

- What is the percentage of fault removal?

Metric(s):

- Fault Removal Rate = No. of corrected faults / No. of total faults discovered

---

Evaluation question(s):

- How many of the required test cases are covered by the test plan?

Metric(s):

- Test Coverage = A /B, where A = No. of test cases executed, B = No. of test cases required

---

Fault tolerance

Fault tolerance of a ubiquitous system refers to its ability to respond gracefully to an unexpected hardware or software failure. There are many levels of fault tolerance, the lowest being the ability to continue operation in the event of a power failure. Many fault-tolerant computer systems mirror all operations -- that is, every operation is performed on two or more duplicate systems, so if one fails the other can take over. The following questions and measurements should be considered for evaluation purpose.

Evaluation question(s):

- How many fault patterns were brought under control to avoid critical and serious failures?

Metric(s):

- Fault avoidance rate = A /B, where A = Number of fault patterns having avoidance, and B = Number of fault patterns to be considered

---

Evaluation question(s):

- How many functions are implemented with incorrect operations avoidance capability?

Metric(s):

- Incorrect operation avoidance rate = A/B, where A = No. of functions implemented to avoid incorrect operation patterns, and B = No. of incorrect operation patterns to be considered

---

Fault recovery

Fault recovery of the ubiquitous computing service refers to its ability to re-establish its level of performance and recover the data directly affected in case of a failure and on the time and effort needed for it. The following questions and measurements should be considered for evaluation purpose.

Evaluation question(s):

- How capable is the product in restoring itself after abnormal event or at request?

Metric(s):

- Recovery Rate =A /B, where A = No. of implemented recovery requirements confirmed, and B = No. of recovery requirements in the specification

---

Maintainability

Maintainability refers to the ease with which the ubiquitous computing service can be modified to change or add capabilities, improve performance, or correct defects. Maintainability can be represented by the probability that a failed component will be repaired within a given amount of time. There are four important characteristics of maintainability: analyzability, changeability, stability and testability.
Analyzability of the ubiquitous computing service refers to the ability to diagnose the deficiencies or causes of failures, or for identification of parts to be modified. Changeability of the ubiquitous computing service refers to the ability to make system modification, fault removal or for environmental change. Stability of the ubiquitous computing service refers to the effort to reduce the risk of unexpected effect of modifications. Testability of the ubiquitous computing service indicates the effort needed for validating the modified software. The following questions and measurements should be considered for evaluation purpose.

Evaluation question(s):

- Can user identify specific operation which caused failure?

Metric(s):

- Audit Trail Capability X = A /B, where A = No. of data actually recorded during operation, and B = No. of data planned to be recorded enough to monitor status of software during operation

Evaluation question(s):

- Can user identify specific operation which caused failure?

Metric(s):

- Failure Analysis Capability X = 1- A /B, where A =No. of failures of which causes are still not found, and B = Total No. of registered failures

---

Adaptability

Change is inherent in ubiquitous environments. When a smart device is connected to a ubiquitous network, it immediately starts working with other existing devices and all its functions should still able available. In this case, adaptability of the ubiquitous computing services refers to its ability to adapt to different specified environments without applying other actions or means than those provided for this purpose for the service.

One of the key issues of ubiquitous computing is to support the system adaptability at many different levels. Adaptive systems have high availability requirements and there is a need to model and analyse these architectures during adaptation as well as during normal operation. A compositional approach to modelling and analysis appears to be crucial in dealing with adaptive ubiquitous computing systems.

The adaptability of the system should also be supported by interleaving phases of analysis and refinement of requirements. Software applications and immediately accessible data need to adapt to users' current context in terms of location, activity, device/communicator capabilities. The network environment needs to adapt to provide services for mobile users and particular applications with specific quality of service requirements. Web services need to store and provide a variety of different media. Information sources from multiple distributed heterogeneous databases need to be integrated to form a cohesive information store.

There is a need for self-configuring services and applications that automatically detect and adapt to the resources available to support them; we need distributed programming languages that are extensible, that can be tailored to specific applications and are suitable for producing applications to be embedded in consumer devices.

There are three aspects to be considered for ubiquitous computing system adaptability: Generality, conformance and installability. Generality refers to the capability of the ubiquitous computing service to provide a minimum number of restrictions, so that it may be used in different contexts. Conformance of the ubiquitous computing service refers to the capability of the ubiquitous computing service to adhere to standards or conventions relating to portability, for example, conformance to documentation standards, document readability. Installability of the ubiquitous computing service refers to the ability to install the ubiquitous computing service in a different environment. The following questions and measurements should be considered for evaluation purpose.

Evaluation question(s):

- Can user or maintainer easily adapt software to data sets in new environment?

Metric(s):

- Adaptability of data structure X = A /B, where A = No. of data which are operable but are not observed due to incomplete operations caused by adaptation limitations, and B = No. of data which are expected to be operable in the environment to which the software is adapted

---

Evaluation question(s):

- Can user or maintainer easily adapt application software to environments?
- Is software capable enough to adapt itself to operational environment?

Metric(s):

- Environmental adaptability of hardware X = A /B, where A = No. of operational functions of which tasks were not completed or not enough resulted to meet adequate levels during combined operating testing with environmental hardware., and B = Total No. of functions were tested

---

4.3.3 Contextualisation or Smartness

Smartness – context-aware – services make inference about user's activities, goals, emotional states, and even the social situation, and might attempt to act on the user's behalf (Scholtz and Consolvo, 2004b). If the service has configured and interpreted the context correctly with the support of OS or system software, it can save time and reduce user workloads. Otherwise, the user might have to interdict its actions, potentially resulting in wasted time, embarrassment and even danger. Bellitti et. al., (Bellotti et al., 2002) maintain that the context-awareness system must be intelligible and accountable. However, to make context-aware ubiquitous systems a reality, we need the support of mobile and wireless technologies. For example, IPv6 is a set of network protocol that is capable of controlling a huge number of addresses that are needed to connect diversified sensors.

This is often referring to smart house or smart room, where small sensors, such embedded processors are used to detect their surroundings and equip human with both information processing and communication capabilities.

For instance, a smart house is a truly interactive house using the latest information and communication technology to link all the mechanical and digital devices available today. For example, humans could easily find out where they were, what other objects were in their vicinity, and what had happened to them in the past. They could also communicate and cooperate with other "smart" objects and, theoretically, access all sorts of Internet resources. Objects and appliances could thus react and operate in a context-sensitive manner and appear to be "smart," without actually being "intelligent." (Friedemann, 2001)

Three aspects of context-awareness (or smartness) should be considered for evaluation purpose: Location awareness, environment awareness and situation awareness.

Location Awareness

Today's computers have little idea of their location and surroundings. In a Ubiquitous Computing environment, a computer knows what room it is in and it adapts its behaviour in accordingly. This allows for an improved interface between the user and many applications, as the computer system can be more aware of the context in which the user is operating.

Ubiquitous computing removes location dependency and instead makes conscious awareness of users' location. By definition, a location-aware ubiquitous computing system is one that makes use of contextual information (such as location, orientation, etc.) to facilitate a human centred information environment that is familiar and known to the user. A location-aware ubiquitous system should provide services discovery functions triggered by user location changes where a corresponding service is automatically found upon a request based on best match. The following questions and measurements should be considered for evaluation purpose.

Evaluation question(s):

- How accurate can the ubiquitous computing system sense the user location?

Metric(s):

- Location Sensitivity = confidence intervals *($\alpha \pm \beta\%$)* in terms of the real location of the user

---

Evaluation question(s):

- What is the relative Service Discovery Rate based on user locations?

Metric(s):

Service discovery rate = A / B, where A = No. of successful discoveries based to user locations

B = No. of total discoveries based to user locations

Environment awareness

Environment awareness takes ubiquitous computing beyond simple location awareness, by provides the system not only with details of its current location, but also environmental factors, such as temperature, humidity, ambient light level, etc.

An environment aware mobile computer system is one that makes use of environmental information to facilitate a human centred information environment that is familiar and known to the user. The following questions and measurements should be considered for evaluation purpose.

Evaluation question(s):

- Is the system capable of providing environment-aware, such as location-aware and orientation-aware sensor data?

- Is the system capable of interacting with the user as he/she wishes?

Metric(s):

- Environment Sensitivity = confidence intervals *(α±β%)* in terms of the real environment of the user

---

Evaluation question(s):

- What is the relative Service Discovery Rate based on user environments?

Metric(s):

Service discovery rate = A / B, where A = No. of successful discoveries based to user environments
B = No. of total discoveries based to user environments

<u>*Situation awareness*</u>

Research reported from different communities, such as Wearable Computing and Mobile Computing, indicates that awareness of situations can lead to improvement of human-computer interaction. An situation-aware ubiquitous system makes use of situation information to facilitate a human centred information environment that is familiar and known to the user.
Situation information obtained by the ubiquitous system can be measures of the motion of the user, the surrounding light conditions, the orientation of a display, users' relative position to a device, the number of users in front of a device, users' emotional state (bio-sensors), etc. This situational context can be captured and used as additional input to the system (Abowd et. al, 1998).
Situation awareness takes ubiquitous computing beyond simple location awareness, by provides the system not only with details of its current location, but also environmental factors. The following questions and measurements should be considered for evaluation purpose.

Evaluation question(s):

- Is the system capable of providing environment-aware, such as location-aware and orientation-aware sensor data?
- Is the system capable of interacting with the user as he/she wishes?

Metric(s):

- Situation Sensitivity = confidence intervals *(α±β%)* in terms of the real situation of the user

---

Evaluation question(s):

- What is the relative Service Discovery Rate based on user situations?

Metric(s):

Service discovery rate = A / B, where A = No. of successful discoveries based to user situations
B = No. of total discoveries based to user situations

User Recognition

A ubiquitous system should have the smartness of acknowledging the familiar user or user pattern. Emerging technologies like biometric recognition systems make user recognition possible. Biometric recognition systems have been widely deployed in forensic, and other government/commercial applications. Several biometric systems have been developed for automatic recognition of individuals based on their physiological/behavioural characteristics, such as fingerprint, face, voice, signature, keystroke dynamics, etc. The following questions and measurements should be considered for evaluation purpose.

Evaluation questions:

- Can the ubiquitous computing system automatically identify the user?

Metric(s):

- Automatic Identification Rate = A /B, where A = No. of cases the ubiquitous computing system identifies a user correctly, and B = Total No. of cases the ubiquitous computing system needs to identify a user

---

Personalisation

Smart systems might let users customise responses based on personal preferences by explicit input or by letting systems learn and adapt over a series of interaction (Scholtz and Consolvo, 2004b). Personalised services provide a personal touch for the user by selecting specialised content and ads or by defining rules and user segments or even running complex business campaigns. The following questions and measurements should be considered for evaluation purpose.

Evaluation question(s):

- Is the ubiquitous computing system capable of providing personalised information according to user preferences?

Metric(s):

- Personalisation Rate X = A /B, where A = No. of times the ubiquitous computing system provides correct personalised information to a specific user, and B = Total No. of times the ubiquitous computing system need to provide correct customised functions to a specific user

---

4.3.4. Security

The security of ubiquitous computing service refers to establish mutual trust between infrastructure and device in a manner that is minimally intrusive. In an ubiquitous computing environment, a smart device can recognise the user through a sort of Universal Remote Control which the user keeps secured. Secure transient association is used when the user is deploying devices and imprinting can be used to establish shared secret. An example is the mobile banking system which requires extremely high security provided by the ubiquitous computing system.
There are interesting and challenging problems in providing consistency in the management of security and in specifying authorisation policies for ubiquitous computing environments. Security can be implemented in heterogeneous components such as firewalls, different computer operating systems and multiple databases. The ubiquitous computing system should support secure sensitive or high-value transactions and verifies that messages were not modified while in transit from queue to queue. Authentication is one of the most important characteristics of ubiquitous computing security. Authentication provides confirmation of user access rights and privileges to the information to be retrieved. During the authentication process, a user is identified and then verified not to be an imposer. The authentication process is the assurance process that a party to some computerised transaction is not an impostor. The following questions and measurements should be considered for evaluation purpose.

Evaluation question(s):

- How complete is the audit trail concerning the user access to the system and data?

Metric(s):

- Access Auditability $X = A / B$, where A = No. of "user accesses to the system and data" recorded in the access history database, and B = No. of "user accesses to the system and data" done during evaluation

---

Evaluation question(s):

- How controllable is access to the system?

Metric(s):

- Access Controllability $X = A / B$, where A = No. of detected different types of illegal operations, B = No. of types of illegal operations as in the specification

---

Evaluation question(s):

- What is the frequency of data corruption events??

Metric(s):

Data Corruption Prevention Metrics:

- $X = 1 - A / N$, where A = No. of times that a major data corruption event occurred, and N = No. of test cases tried to cause data corruption event

- Y = 1- B/N, where B = No. of times that a minor data corruption even occurred
- Z = A /T or B / T, where T = period of operation time (during operation testing)

Privacy

Protecting the privacy of users is of central importance. In a ubiquitous computing environment, sensors are actively collecting user data, much of which can be very sensitive and valuable. The data collected will often be streaming at high rates (video and audio) and it must be dealt with in real-time (Canny and Duan, 2004). In addition, there could be hundreds of tiny computers in every room, all capable of sensing people near them.

How is privacy maintained when location and activity are tracked (and predicted) by the ubiquitous environment? Imagining that there many computers linked by high-speed networks where messages can be intercepted and recorded by unauthorised people. Effective solutions for controlling access to data in ubiquitous computing environments remain to be a challenge for some time to come.

For privacy, *encryption* would be a favoured way to protect personal data from unauthorised use. Encryption is the transformation of data into a form unreadable by anyone without a secret decryption key. Its purpose is to ensure privacy by keeping the information hidden from anyone for whom it was not intended, including those who can see the encrypted data. Encryption may be used to make stored data private (e.g., data that is stored on a potentially vulnerable hard disk), or to allow a non-secure communications channel to serve as a private communications channel. Encryption is sometimes described as the process of converting plain text into cipher text.

Privacy is one of the key characteristics of the ubiquitous computing service However, ubiquitous computing systems should be built to have the same privacy safeguards as the real world, but no more, so that ethical conventions will apply regardless of setting. The following questions and measurements should be considered for evaluation purpose.

Evaluation question(s):

- Is data encrypted to protect the wearer’s privacy?
    - To prevent the wearer's identity being discovered
    - To prevent the wearer's personal details (sensor data) from being discovered
    - To prevent the wearer's habits from being discovered
    - To prevent impersonation
    - To Provide encryption?

Metric(s):

- Data Intrusion Prevention Metrics:

    - X = 1- A/ N, where A = No. of times that a major data intrusion event occurred, and N = No. of test cases tried to cause data intrusion event
    - Y = 1- B/N, where B = No. of times that a minor data intrusion even occurred
    - Z = A /T or B / T, where T = period of operation time (during operation testing)
- Types of information user has to divulge to obtain value from services
- Availability of the users information to other users of the system

---

4.4 Systems Software and Networks

System software is built to support change, which is inevitable. Change is occasioned by anonymous devices customising to users, by explicit user requests, by the needs of applications and their components, by current operating conditions, by the availability of new software and upgrades, by failures, or by any number of other causes.
Ubiquitous system software architecture must support control and planning abstractions that provide mechanisms for change, on specifications that support putting these mechanisms to use, and on persistent object stores with transactional semantics to provide operational support for change.
Ubiquitous computing device hardware is not simply limited to handheld devices. There is also a large amount of fixed-infrastructure hardware that may be incorporated into a system. The software that runs the hardware devices, and co-ordinates the usage of the physical resources is of paramount importance. In this section we describe the measurable aspects of the systems software (managing hardware resources) and also the intercommunication networks.
The ubiquitous computing systems software and networks have two orthogonal aspects: configuration and also co-ordination.

4.4.1 Configuration

The systems software and networks used by a Ubiquitous computing system need to be able to be adequately configured so that the ubiquitous computing system can effectively use the hardware. From a purely configurational point of view, the most important aspects of configuration management are: Scalability, Security and Extensibility.

Scalability

A system is defined as being scalable if the overhead required to add more functionality is less than the benefit that functionality provides. Thus if adding a server to the network makes it more difficult to accomplish a task, the system is not scalable.
For instance, Jini addresses scalability through federation. Federation is the ability for Jini communities to be linked together or federated into larger groups. The ideal size for a single Jini community is a workgroup, consisting of a number of printers, scanners, PDA’s and network devices required by a group of 10 to 100 people. Jini groups can then be brought together in a community in order to make their resources sharable.

Access to services within other workgroups is possible through federation. Specifically, the Jini lookup service-the entity responsible for keeping track of all the services in the community- is itself a service. A lookup service of one community can register itself in other communities offering itself as a resource for the community. A Jini federation is self-managed; a device inserts itself into a network. Jini technology dynamically discovers the services it needs for processing client requests. Jini federations are designed to exist in a flexible environment thus making them more scalable.

Scalability describes the ease with which extra resources can be added to the ubiquitous computing infrastructure. It includes such questions as: what re-configuration of the existing systems must be done when new resources are added into the system (c.f. resource discovery, as discussed in section 4.3). Systems software manages the inclusion of hardware in a ubiquitous computing system; the appropriate software must be available for the hardware that is to be added to the system. Traditionally, software is shipped with 'drivers' or 'Application-Programmer Interfaces' (APIs) that allow technical people to use a new system with existing hardware. Scalable systems are utilised to address the problem of load imbalance in hardware and software systems.

Evaluation Question(s):

- Does the ubiquitous computing system support the ability to add new hardware?
- Does the ubiquitous computing system support the ability to add new software?
- Does the ubiquitous computing system support the ability to add new users?
- Is there a prescribed list of hardware that will work with the ubiquitous computing system? Or what range of hardware is supported by the ubiquitous computing system?
- Is there a list of drivers or APIs that can be used by developers or technical people to use a new piece of hardware with the ubiquitous computing system?
- Does adding new hardware to the system slow down the system?

Measure(s):

- o *List of 'approved' hardware known to work with the ubiquitous computing system*
- Software Drivers or Technical support for adding new hardware to system
- API available to add new hardware to system
- Is there a list of approved hardware to work with the Ubiquitous Computing Environment? How large is the list of approved hardware?
- Is it possible to use a generic hardware description to add new hardware that is not on the list of approved hardware?
- Is there system documentation that describes the correct steps to add new hardware?
- Is it possible to select the protocol used by the system when communicating with remote devices?
- Are users individually known to the system (or are all services run on behalf of an anonymous user)?
- How many steps are required to add a new user to the system?

- How long is it before a new user can access any of the [remote] resources in the system?
- Is there a time at which the peak load is highest? Is the system able to move jobs (see section on Mobility) during these peak times?

Metric(s):

- 1-(A/B) where A is the response time of the system after new hardware is added and B is the response time of the system before the new hardware was added
- 1-(A/B) where A is the boot time of the system after new hardware is added and B is the boot time of the system before the new hardware was added
- 1-(A/B) where A is the response time of the system after a new user is added and B is the response time of the system before the new user was added
- A/B where A is the number of jobs (queries, service invocations) that are waiting to be executed and B is the number of jobs that are concurrently being executed at peak (lower is better).
- A/B where A is the number of jobs (queries, service invocations) that are waiting to be executed and B is the number of jobs that are concurrently being executed on average (lower is better).

Security

One of the fundamental aspects of a hardware device is its level of security. While this aspect is discussed in terms of other metrics (see section 4.3), in this context we refer to the ability for separate user requests or actions not to interfere with those of another. We also refer to the ability for users not to perform unauthorised monitoring on the actions of others. Finally we consider whether the systems software has any infrastructure to protect the physical devices from usage by unauthorised parties.
For instance, Jini tackles this by ensuring that remote clients can be prevented from even seeing what services are offered by the network. Only when they become part of the federation or network are they allowed see its services. This provides a first line of defence. The architecture of Jini also fundamentally protects against Viral attack. Jini servers provide a client the interface with which it may access the services of code residing on a server. The services are accessed through the network and the code itself is typically not transferred. Viruses on the other hand for infection, typically require code mobility and use this mobility to transfer executables and infect other computers. Another example is the use of mobile telephones for personal banking. The security of such a system would be found on two levels: the application level and the system level. At the application level, some form of authentication and authorisation protocols would have to be put into place to ensure that users are who they say they are, and that messages cannot be 'spoofed'. At the system level, actual data transmissions would likely have to be encrypted to prevent interception and monitoring.

Evaluation question(s):

- Does the system software logically separate user requests?
- Does the system enforce this logical separation?
- Does the system isolate a failure in one user's request from another?

- Does the hardware have any intruder protection (i.e. password security, biometric protection)
- Does the system support protection against [Distributed] Denial of Service attacks?

Measure(s):

- System remains stable when another user's job fails
- Users unable to inspect actions of other users
- Ability for unauthorised users to interact with system – locally or remotely
- Ability to control which users have access to which services on each and every piece of hardware within the ubiquitous computing system
- Does there system require the presence of non-trivial passwords or biometric access control for devices?
- Can the hardware be secured (i.e. lockable cabinets, etc)?
- Are users required to authenticate themselves before being able to use the Ubiquitous Computing Environment?
- Are users required to re-authenticate themselves periodically when using the Environment?
- Do some devices or services require additional authentication from users?
- Does the Environment support differing levels of user class (e.g. User, Administrator, etc.)?
- Are attackers able to determine the presence of data in transit from radiated signals on hardware devices?
- Is encryption used to secure data stored on hardware devices?

Metric(s):

- A/B where A is the number of jobs that fail when another users' jobs fail and B is the total number of jobs running
- A/B where A is the number of other users' jobs that are running that another user can 'see' and B is the total number of other users' jobs running
- A/B where A is the number of services that an unauthorised/unauthenticated user can start without authorisation/authenticating themselves and B is the total number of services available to authorised users
- A/B where A is the number of physical devices that have been secured and B is the total number of physical devices
- A/B where A is the number of physical devices upon which data is stored in an encrypted form, and B is the total number of physical devices upon which data is stored in the system

---

Network Security

As networking trends move toward ubiquitous structuring schemes, a constant limitation on its success and growth is security. For example, unwarranted access of ubiquitous network can give opportunity to a malicious party to perform changes to the infrastructure allowing for larger-scale attacks. The ubiquitous computing paradigm has

created new challenges in security and requires that we develop new approaches to address both existing and new security problems.
Intercommunication networks must also exhibit the qualities of security. The networks, themselves, must be beyond physical tampering: network devices such as switches and hubs must be secure; any wires, fibre or airwaves must be secure and also the traffic traversing the networks must not be able to be tampered with.

Evaluation question(s):
- Are network devices physically accessible?
- Are there any guarantees that there are no eavesdroppers on the network media
- Are there secure protocols in place to guarantee end-to-end authentication and authorisation?
- Are there feedback mechanisms to detect unauthorised usage attempts?

Measure(s):
- Are the network devices physically secure?
- For wireless networks, are non-obvious (non-default) WiFi encryption keys used?
- Do network switches and other hardware have their default passwords for administrative access changed to something non-trivial?
- Does the system inform administrators when hacking attempts have been recorded?
- Which [secure] protocols are supported by the system and network devices?
- What authentication/authorisation is required to change secure network device passwords or configurations? Who is able to do this?
- Are attackers able to determine the presence of data in transit from radiated signals on transmission media?
- Is encryption used to secure data stored on network devices and/or in transit on networks?

Metric(s):
- A/B where A is the number of network devices with non-trivial (non-default) passwords set, and B is the total number of network devices
- A/B where A is the number of hacking attempts reported to administrators and B is the total number of hacking attempts on network devices
- A/B where A is a measure of the amount of network traffic (data) that is encrypted and B is the total amount of network traffic over a given time period

---

Extensibility

Once hardware has been added into the Ubiquitous Computing environment, can its functionality be extended beyond the level that was originally supplied on purchase of the system? This also refers to whether the system can be upgraded in the field once deployed to customers.

Evaluation question(s):

- Is there a programmer API that can be used to extend the system?
- Can programmers interrogate the services or other hardware used by the system?
- Can new programmes create new events within the system ("value-adding" to the original)
- Are the programmes created by programmers able to be 'discovered' by the resource discovery system (see section 4.3)?
- Can customers perform software upgrades to the system in the field?

Measure(s):

- Is there a published API and documentation for programmers to extend the system?
- Is there technical support available for programmers?
- Are new programmes able to be used in the same way as those supplied by the manufacturer of the ubiquitous computing system?
- Is there example code available for applications programmers?
- How many steps are required to investigate the status of local hardware and/or services
- How many steps are required to investigate the status of remote hardware and/or services
- How complex is it to add a new software component to the system?
- Is there an option in the system software to apply upgrade patches in the field?
- Is it possible to 'back-out' of software upgrades performed in the field should they prove unsuccessful?

Metric(s):

- A/B where A is the number of software and hardware systems that can be upgraded in the field and B is the total number of hardware and software systems in the environment that could need upgrading
- A/B where A is the number of software systems that have technical support available from the Vendor and B is the total number of software systems in the environment
- A/B where A is the number of hardware systems that have technical support available from the Vendor and B is the total number of hardware systems in the environment
- A/B where A is the number of remote devices that can have their status polled remotely and B is the total number of remote devices in the environment
- A/B where A is the number of times new software can be added to the system successfully and B is the total number of attempts that are necessary to successfully add the software to the system

4.4.2 Co-ordination

Co-ordination describes the system software's ability to co-ordinate a user request operating across the ubiquitous computing system. We sub-divide this category into Resource Co-ordination, Mobility, Reliability and Distributed Systems support.

Resource Co-ordination and Optimisation

How effectively does the systems software control the resources that are specifically attached to this hardware. In the context of Ubiquitous Computing Environments, a resource consists of any physical device (passive or active sensor, mobile device, fixed-location computing or network device) or software that is potentially shared between multiple users. For example, in the case of a desktop computer, does it allow concurrent user access? Does it support remote access to local resources?

Evaluation question(s):

- How well does the systems software control resource sharing on the local hardware

Measure(s):

- Is multi-user operation supported?
- What scheduling algorithm is used to ensure fairness amongst concurrent user operations
- Does the systems software enforce resource limits (such as the amount of memory or disk space) that a user request can consume?
- What is the behaviour of the system as resources on the local machine become low?
- What is the behaviour of the system as resources on remote machines become low?

Metric(s):

- 1-(A/B) where A is the response time of the system after a new service is invoked and B is the response time of the system before the service was invoked.

Mobility

Mobility support describes the ability for system software to recognise that programs (services) or data would be more efficiently located at a remote machine. Systems that support mobility might have the ability to move code or data to a remote machine. In sophisticated systems there may also be the ability for a system to move a *running program* to a remote machine to increase its efficiency; this is the concept of load-levelling across distributed computational resources. Also, when a physical hardware device is moved, how quickly does the ubiquitous computing system recover to the point where other components of the wider system are able to use the moved component? This topic is also strongly related to the issue of Portability (see Section 4.3).

Evaluation question(s):

- Is the resource monitoring aspect of the systems software sophisticated enough to judge whether a program or code would be more efficient if located elsewhere?
- Is there enough information available on remote resources to make a good judgement on where to locate code/data

Measure(s):

- Can systems software re-locate a program or piece of data to improve efficiency
- Is a sensible decision made as to the destination location of a moved code/data
- Is hardware is moved, how quickly can it be 'seen' and used by other parts of the ubiquitous computing system?
- Are standard protocols supported that make it possible to use different underlying hardware, such as transmission media (wireless networks, optical fibre networks or traditional cabled networks).

Metric(s):

- 1-(A/B) where A is the time for the system to locate a re-located piece of hardware and B is the time to locate the hardware before moving
- 1-(A/B) where A is the time for the system to access a re-located piece of hardware (after the system is quiescent) and B is the time to access the hardware before moving

---

Reliability

Reliability is the measure of "how well a device or network performs in the presence of disturbances". However processes often crash, or someone will shut them down. Or the machine running the process will crash or stop. This aspect of systems software and communication networks covers the dual points of fault tolerance and also dependability. Fault tolerance describes the ability of the system to continue working in the presence of faults. Faults can be of many types, caused by many levels of hardware and software. Ideally, the system should remain operational in the presence of faults, and inform some nominated party so that the fault can be rectified.

Dependability describes the property of the system to be Available, Reliable, Safe and Maintainable (Tanenbaum and van Steen, 2002). The system should be available for the periods for which it is meant to be accessible by the ubiquitous computing system; it should be reliable in that it should execute, correctly, any valid user request submitted to the system; the execution of a user request should not cause the system to produce un-safe (i.e. dangerous) side-effects whether the request can be executed successfully or unsuccessfully. Finally, the system must be maintainable: the replacement of a faulty component, or the isolation of a faulty component should not simply result in the replacement of the whole piece of hardware.

For instance, Jini is capable of managing these changes due to the fact that it expects devices to randomly move in and out of the network. In Jini when a server becomes unavailable, the client automatically goes looking for an alternate server. Once it locates another server process (or the failed process comes back up), the Jini client can reconnect. If no server is available, the client waits or informs the user. These features combined make Jini virtually unique among commercial-grade distributed systems infrastructures. They make a Jini community virtually administration-free.

Evaluation question(s):

- Does the system continue working in the presence of a variety of faults?
- What technical support is available for the system software or hardware?

Measure(s):

- What isolated faults cause the system to stop responding?
- What combinations of faults cause the system to stop responding?
- Is there technical support available for the ubiquitous computing system? The hardware?
- How many steps must be taken to diagnose the cause of a hardware fault in a local machine?
- How many steps must be taken to diagnose the cause of a hardware fault in a remote machine?
- Can the system produce any dangerous side-effects in the presence of faults?
- In the presence of un-related faults, do user processing requests still complete in a timely fashion?

Metric(s):

- What is the availability (‘up-time’) of the hardware? 1-(A/B) where A is the time that the hardware has been unavailable due to fault(s) and B is the sum of the time the hardware has been operating and to has been unavailable
- What is the availability (‘up-time’) of the systems software? 1-(A/B) where A is the time that the systems software has been unavailable due to fault(s) and B is the sum of the time the systems software has been operating and to has been unavailable.
- What is the availability (‘up-time’) of the network switching devices? 1-(A/B) where A is the time that the network hardware has been unavailable due to fault(s) and B is the sum of the time the network has been operating and to has been unavailable
- What is the mean time between failures (MTBF) for the infrastructure hardware in the system? (see section 4.3)
- What is the mean time between failures (MTBF) for the infrastructure network switching hardware in the system? (see section 4.3)
- What is the mean time between failures (MTBF) for the infrastructure systems software? (see section 4.3)

Distributed Systems Support

Support for Distributed Systems describes whether or not the systems software is 'aware' that it is part of a larger system. For example, many desktop (and mobile) hardware operating systems are not aware that they may be part of a larger system; any distributed systems aspects of these systems must be provided by extra applications. In a tightly-coupled distributed operating system or middleware layer, many hardware devices may be dependent on the availability of distributed resources. For example, in OSF's DCE, there must be a Security Server, a Time Server, and a Directory Server available all the time; if one of these servers is not available to other members of the distributed system, users are unable to submit processing requests.

Evaluation question(s):

- Is the systems software a 'distributed/network operating system' or a 'middleware layer'?
- To what level are the distributed physical resources inter-dependent?

Measure(s):

- Is the systems software aware of being part of a larger network with shared resources and multiple users?
- How does the failure of a remote machine affect the performance of other hardware in the ubiquitous computing environment?
- How many machines must still be operating for users to be able to submit new jobs?

Metric(s):

- o 1-A/B where A is the number of machines that have been booted in the environment and users are still able to submit new jobs (and have them complete) and B is the total number of machines in the environment
- o 1-A/B where A is the number of machines still operating (in the presence of other machines being unavailable) and users are still able to submit new jobs (and have them complete) and B is the total number of machines in the environment

5 Case studies: Application of the Framework

We have discussed what scopes have to be considered in evaluating the ubiquitous computing system. It has classified the quality characteristics in terms of four different categories, say users, I/O devices, ubiquitous service, and ubiquitous system software, along with corresponding quality evaluation questions. However, the framework proposed is mostly based on the literature in this domain, not resulting from any empirical investigation on the industrial setting. Therefore, we are very carefully claiming that the evaluation scopes and the quality metrics are not fully applicable in the current industrial setting. Albeit this issue, this research is promising many implications to the current laboratory or industrial ubiquitous computing systems.
In this chapter, we will investigate how equally the evaluation scopes we have proposed and the quality characteristics could be matched with the current technological feasibility of the ubiquitous computing systems, so we can see what benefits and the deficiencies of the proposed framework would be. Three examples are considered: Portable Help Desk, Vodafone™ Germany location-awareness service, Telecom™ NZ and SK telecom™ China mobile banking service. These three applications can be seen as the most-likely applications in the current ubiquitous computing environment.

5.1 Portable Help Desk

Portable Help Desk (PHD: Smailagic et al., 2000) developed a campus tour guide that tracks user location and provides information about surroundings. Developed for visitors, it displays a map on a PDA.
Whilst the development is still underway, Smailagic at al., have evaluated the system, focusing on the features, the balance of attention between the device and the physical environment, and context awareness service functionality. Here we list several of their findings, suggesting in parentheses appropriate evaluations scopes from our framework and their related metrics:

- The devices were distracting. Users looked at the devices, not local surroundings. *(Expression- Awareness support, Unobtrusiveness)*
- Users wanted more functionality. For example, direction to other places on campus were needed but unavailable form the system *(Ubiquitous service - Functionality, Expression- Controllability)*
- The application did not always sense users location accurately, which caused confusion about the building actually described in their PDA devices *(Reception- quality of reception, Contextualisation – location awareness)*

Most of these findings are based on the laboratory version of the system, so there is less likely to result in service resource availability, security, and scalability, which are the main concerns of the industrial setting. The following two case studies are more concentrated on the difference between the industrial setting and the laboratory situation.

5.2 Vodafone Germany location-awareness service

Mobile phones have spread faster than any previous technologies for which figures exist. While mobiles are sometimes seen as a fashion term, their personal benefits go well beyond the fashionable. Namely, they save time (doing business on the move),

provide security (emergency services are close at hand), provide entertainment (games and pictures) and most of all, keep us in touch with family and loved ones.
As one of the planned market development, Vodafone Germany (2000) introduced one of the first location awareness services in the world. For instance, the service has offered that the user could request traffic information, shopping information and hotel guide information relevant to her current location via the WAP portal of Vodafone. Therefore, a user accessing the hotel guide did not have to explicitly input the city where she was currently, but was localised automatically by the operator via the base station her mobile phone was currently logged in. This sort of ubiquitous services should follow some challenges to be met, suggesting in parentheses appropriate evaluations scopes from our framework and their related metrics:

- All location-based services offered so far to mobile phone users are always pull-services, say, the user has to explicitly request information relevant to her current location. This is particularly annoying if the user always has to establish a WAP connection before he can use such services. It takes a lot of time, which limits the values of the service to the user, and it is also very costly as the user usually has to pay for the online time. Thus, the service should be very easy to use, therefore must offer a simple, yet powerful interface, be robust, and error-tolerant, covering the reasonable service region *(Simplicity – Activity, Reliability – Service quality and Coordination, Service resource availability – Service quality).*
- The tracking of the location, which is necessary to provide the location awareness, should be transparent to the user with regard to the system is monitoring the whereabouts of the users, so she understands how her activities would make effects on the system *(Transparency – Expression)*. This is closely linked to the privacy issue as follows.
- One of the main advantages of the location-awareness service is to help the service users not only improve productivity, but also to help ensure the safety and security of them, if they are frequent mobile workers *(Quality of reception, health and safety issue – Reception)*. On the other hand, the location-awareness service must be secure and private to the users. As a worst case-scenario, if some unauthorised people can see the whereabouts of the users, it could result in a “big-brother syndrome” in the society *(Security and Privacy)*. Following on the privacy issue, the service must ask to the user of approval or overt acknowledgement if the system is monitoring the user *(Acknowledgement – Evaluation, Awareness support, Unobtrusiveness – Expression).* Therefore, the user should be able to turn on/off the service to actively confirm her wish for localisation for privacy reasons *(Privacy and Security, Controllability - Expression)*
- The most important technical concern of the service is accuracy. For instance, the localisation based on Vodafone GSM cells allows an accuracy of about 100 to 500 metres in the city. In contrast, a recent advanced system such as Cambridge Positioning Systems (CPS) is the high accuracy location enabler, helping people pinpoint their whereabouts - via a standard GSM mobile handset around sub-100m accuracy with rapid time-to-fix and consistent performance across all environments *(Location awareness – Contextualisation, Quality of*

reception - Reception). In this service, the user always has to explicitly request the information. However, in order to realise the proper location awareness service, information has to be automatically delivered to the user *(Functionality, Adaptability – Services, Unobtrusiveness - Expression)*

- The system architecture for tracking location should also not be restricted to or depend on specific kinds of trackers. It must allow adding new trackers (perhaps even newly developed trackers) and including them easily in the overall architecture *(Extensibility, Scalability – Configuration)*.

5.3 Telecom NZ and SK telecom's mobile banking service

As we have suspected above, the main applications of the ubiquitous service, of course, would be via mobile phones. This is partly the mobile telecommunication technologies offer a mostly saturated market environment, and partly because the major vendors have many technical advances to support the environment.
Recently, SK telecom (2004) announced that the company is introducing the mobile banking service in a joint effort with Woori™ bank in China and Korea. This service allows customers to enjoy mobile banking service through a wireless internet service. This ubiquitous banking service, first launched in 2002, let customers use a variety banking service with a single IC (Integrated chip). For instance, customers can make enquiries about their cheques, bank account balances, and banking transaction and so on, as well as transmit money to other banks or financial solution. According to their plan, this service will be extended to New Zealand via Telecom NZ and Vodafone™ NZ in 2004. Here we list several of the quality characteristics to be guaranteed, suggesting in parentheses appropriate evaluations scopes from our framework and their related metrics:

- The customer can use the service anywhere that the mobile phone service in New Zealand, such as Telecom's CDMA 027 and Vodafone's network provides coverage, offering broad coverage throughout New Zealand. There are areas which the user will receive a stronger signal than others such as metropolitan areas and suburbs. Also, in some instances, you may find small patches where the signal is minimised due to features such as hills, valleys, trees, tunnels, or buildings which may distort or block the signal. By contrast, Vodafone's global network allows the mobile banking to be sued across New Zealand and in over 100 countries around the world using Vodafone's roaming service *(Quality of reception – Reception, Maintainability – Service quality).*

- One of the major banks in New Zealand, ASB bank, has introduced mobile banking service, taking security seriously and offering the security of a secure 128 bit encrypted connection. Furthermore, the customers are required to enter their personal access code and passwords to sign on the system *(Security – Service and Configuration)*. They also provide some emergency situation control such as battery discharges while doing banking service. All of the banking transactions the customer made before their phone battery discharged should have dong through. Once customer's battery is charged and checks the balances and the record of the last transactions the customers made through the statement service. Alternatively, because all the online service are real time, the customers can also check the balanced and last transactions through internet banking or telephone banking service *(Reliability – Service and Coordination)*

- This service can only be accessed through Telecom CDMA WAP service on their 027 network or through a Vodafone approved WAP enabled mobile phone on the Vodafone network. Therefore, the WAP interface of the service should provide ease-of-use in the usability terms *(Simplicity – Activity, Controllability – Expression)*. In addition, as the customers' accounts will be displayed on the screen of the mobile phone, so the customer can access their accounts quickly without fuss when they are on the move *(Simplicity – Activity, Functionality – Services, Mobility – Reception)*.
- The service is designed to cope with a large number of users and has a back-up system *(Scalability – Configuration)*. However, in exceptional circumstances, the service may become unavailable while the users are using the system. Therefore, once the mobile banking service is up and running again, the customer can sign on and check your account balances and the record of the last transaction the customer made *(Reliability – Service quality, Co-ordination)*, or be easily accessible to the staff help support from the bank (Help support – Accountability)

In conclusion, the framework we have developed will be refined as we and others use it. Also we will need to answer interesting questions about relationships among the quality characteristics in the industrial setting.

References

Abowd, G.D., Atkeson, C.G., Brotherton, J.A., Enqvist, T., Gully, P.A., Lemon, J. (1998) *Investigating the capture, integration and access problem of ubiquitous computing in an educational setting*. In the Proceedings of CHI '98, pp. 440-447.

Apple computer (1992) *Macintosh Human Interface Guidelines,* Addison-Wesley, Reading.

Bellotti, V. M. E., Back, M., Edwards, W. K., Grinter, R. E., Henderson, A. and Lopes, C. (2002) *Making sense of sensing systems: five questions for designers and researchers*, *CHI*, 415-422.

Canny, J. and Duan, T (2004), *Protecting User Data in Ubiquitous Computing Environments: Towards Trustworthy Environments*, Privacy-Enhancing Technologies (PET) 2004, Toronto, CA, May.

Crook, C. and Barrowcliff, D. (2001) *Ubiquitous computing on campus: patterns of engagement by university students*, *International Journal of Human-Computer Interaction,* **13,** 245-256.

Davies, N. and Gellersen, H.-W. (2002) *Beyond prototypes: challenges in deplying ubiquitous systems*, *IEEE Pervasive Computing,* **January-March**.

Friedemann, M. (2001) *The Vision and Technical Foundations of Ubiquitous Computing*, *Upgrade,* **2,** 2-6.

Geihs, K. (2001) *Middleware Challenges Ahead*, *IEEE Computer*.

Gorlenko, L., Merrick, R. (2003), *No wires attached: usability challenges in the connected mobile world*, IBM Systems Journal, December 2003.

Hawick, K. A., James, H. A., Silis, A. J., Grove, D. A., Kerry, K. E., Mathew, J. A., Coddington, P. D., Patten, C. J., Hercus, J. F. and Vaughan, F. A. (1999) *DISCWorld: An Environment for Service-Based Metacomputing*, *Future Generation Computer Systems,* **15,** 623.

Hegering, H-G., Kupper, A., Schiffers, M., Buchholz, T. (2002) *Realizing location-based push-services on mobile devices*, Technical report. University of Munchen

Jameson, A. (2003) In *"Adaptive Interfaces and Agents"* Lawrence Erlbaum Association, pp. 316-318.

Lauff, M. and Gellersen, H. W. (2000) *Adaptation in a Ubiquitous Computing Management Architecture*, *ACM Symposium on Applied Computing***,** 566-567.

Lave, J. and Wenger, E. (1991) *Situated Learning: Legitimate Peripheral Participation,* Cambridge University Press, New York.

Lopes, C. and Aguiar, P. (2001) *Aerial Acoustic Communications*, *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*.

Maffioletti, S. (2001) *Requirements for an Ubiquitous Computing Infrastructure*, *Ubicomp*.

Mankoff. J., Dey, A.K., Hsieh, G., Kientz, J., Lederer, S., Ames, M. (2003) *Heuristic evaluation of ambient displays, CHI 2003*

Naghshineh, M. and Willebeek-LeMair, M. (1997) In *IEEE Communications Magazine*, pp. 72-81.

Nahrstedt, K. (2000) *Distributed QoS Compilation and Runtime Instantiation*, *IEEE/IFIP International Workshop on Quality of Service***,** 198-207.

Nahrstedt, K. (2001) In *IEEE Communications Magazine*.

Ryu, H. and Monk, A. (2004) *A brief account of interaction problems*, *OZCHI*.

Scholtz, J. and Consolvo, S. (2004a) *Toward a displine for evaulating ubiquitous computing applications,* Intel Research.

Scholtz, J. and Consolvo, S. (2004b) In *IEEE Pervasive Computing*, pp. 82-88.

Smailagic, A., Siewiorek, D. P., Anhalt, J. and Gemperle, F. (2000) *Towards context aware computing expereinces and lessons*.

Smailagic, A., Siewiorek, D. P., Anhalt, J. and Gemperle, F. (2001) *Towards context aware computing expereinces and lessons*, *IEEE Journal on Intelligent Systems,* **16**.

Suchman, L. A. (1987) *Plans and Situated Actions: The Problem of Human-Machine Communication,* Cambridge University Press, New York.

Tanenbaum, A.S., Maarten van Steen (2002) *Distributed Systems Principles and Paradigms,* Prentice Hall.

VeriTest (2002) *VeriTest-TTA Logo Program: Application Test Check List*.

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

Weiser, M. (1998) *The Future of Ubiquitous computing on campus*, *Communications of the ACM,* **41,** 41-42.