State of the Art: Context Management

Keara Barrett
Telecommunications Software Systems Group
Department of Informatics, Mathematics and Physics
Waterford Institute of Technology

Ruaidhri Power
Knowledge and Data Engineering Group
Department of Computer Science
Trinity College Dublin

1. Introduction

The main objective of ubiquitous computing is to decrease the effort required to exploit computing to aid human activities.

> "Ubiquitous computing has as its goal the enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user" (Weiser 1993).

Furthermore if the user wishes to extract the maximum benefit from the computing environment, the associated systems and services must cooperate and integrate their information and the general information about the situation in which the user wishes the carry out the task. This information about the user's situation is called context. Humans have always used situational information, or context, to make inter-personal interactions richer. If computers use context while interacting with humans, they can offer more useful services and information to humans than is possible without the application of context. The main challenge with computers using context while interacting with humans, is that there is no standard, reusable model that can be used to handle context.

Computers are separated from the reality around them, limited to the explicit input that they receive from their environment. This can lead to differences between what the computer attempts to do and what the user wishes to do, because the computer may not be given enough information to recognise what the user aims to do. Giving computers more information about the world in which we live and work will enable them to assist in our daily lives. The field of context management attempts to address the difficulty of using context in computing by proposing context information that can be extracted from different computing situations. By supplying context information to applications involved, the user experience can be enhanced and new applications can be produced.

The main propose of this paper is to investigate the use and management of context in a system and to examine the components needed to create a proficient context management system. This paper introduces the notion of context, the element of a context-aware system and context management. Benefits and possible uses of context are described and an overview of some existing context-aware systems is given. The remainder of the paper is organised as follows. Section 2 presents an overview of both context and context-aware computing. In Section 3 context is classified, characteristics of context are outlined, elements and features of context-aware computing and context-aware systems are discussed and management models are outlined and compared. Section 3 also discusses some uses of context, it examines privacy as it applies to context, and finally examples of context-aware systems are outlined. The paper concludes with Section 4, which talks about future work in the area of context management.

1.1 Relevance to M-Zones

The main objective of the M-Zones project is to "undertake fundamental research into novel management infrastructures to enable collaboration and management, between and within Smart Spaces." Smart Spaces are work environments with embedded computers, information appliances, and sensors allowing people to perform tasks efficiently by offering unprecedented levels of access to information and assistance from computers

One of the areas of research that is vital for the collaboration and management between and within Smart Spaces is the organisation of context information, which involves the gathering, interpretation, storage and dissemination of context information dynamically and in real-time. The organisation of context information, otherwise called context management, should allow for context-aware services in smart space environments. This should entail the seamless accessibility to context-aware services as the entity, be it a person or device, moves between different smart spaces.

The challenge for the M-Zone project is to examine context and the progress that has been made thus far in the field of context management to recognise the characteristics that a context management standard should include. When the characteristics have been identified they should be integrated into the overall inter-smart space management framework. At present no context management standard exists, so the M-Zones project should research the characteristics of context to recognise the essential aspects of context management.

The M-Zones project should also consider the dynamic exchange of context information between, private, semi private and public smart spaces administered by different parties, to identify the different constraints and effects that mobility and security have on the management of context information.

2. Overview

Much debate has occurred and is still taking place about the meaning of both context and context-aware computing. Therefore one of the first steps in context management is to determine what information constitutes context. Schilit and Theimer, the pioneers of context-aware computing, regard context to be location, identities of nearby people and object, and changes to those objects. They consider where you are, whom you are with, and what resources are nearby to be the important aspects of context. Abowd et. al.'s more recent classification of context (Abowd, Dey, Orr, & Brotherlon 1997) expands the Schilit et al definition. They define context as:

"…any information that can be used to charaterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and application themselves."

This means that any information that depicts the situation of a user can be entitled context. The temperature, the presence of another person, the nearby devices, the devices a user has at hand and the orientation of the user are examples.

By incorporating the "five W's" into the examination of context we can clarify the scope and importance of context in pervasive computing. The questions to be asked include: What is context? Who might benefit from an awareness of their context, whose context is important to who, or what? Where can an awareness of context be exploited? When is context-awareness useful? Why are context-aware applications useful? (Morse, Armstrong & Dey 2000)

When humans talk directly to one another they are able to use implicit information about the situation, i.e. context, to enhance the conversational bandwidth. Unfortunately this implicit information does not transfer naturally to human-computer dialogue (Dey 2002). The concept behind context-aware computing is to exploit the progress in sensing and mechanisms for observing the environment to systematically collect implicit context (Abowd et al 2002).

Context information can be formed into an abstract model of all the actors in a smart space system. A system is context-aware if it uses the context information in the abstract model to provide relevant information and/or services to the user. Services consist of an application and the composition of that application for the task in hand in the system. Context-aware systems can make more informed decisions about information to be presented to users and how to react to commands received from users. According to Schilit and Theimer context-aware computing is any system that "adapts according to its location of use, the collection of nearby people and objects, as well as changes to those over time" (Schilit, Adams & Want 2002)

3. Analysis

3.1 Classes of Context

Context is effective only when shared (Winograd 2001). To ensure context is shared, context must first be gathered and managed by a context-aware system. This implies that the context-aware system must understand what context is before it can go about seeking and categorising this information. Schilit et. al. propose the following classification of context information: (Schilit, Adams & Want 1994)

• *Computing Context* - network connectivity, bandwidth, and nearby resources such as printers, displays, or workstations.

• *User Context* - the user's profile, location, nearby people, and current social situation.

• *Physical Context* - lighting, noise level, traffic conditions, temperature.

Each of these categories contains a wealth of information relevant to the context-aware system. They cannot however be used in isolation to full effect; computing context will be combined with user context, and user context will be combined with physical context to provide a full picture for the context-aware system.

3.1.1 Physical and Virtual Context

Context as defined in section 2 may alternatively be subdivided into two categories: Physical Context and Virtual Context. The intention of a context-aware system is to gather both physical and virtual context and merge them together to achieve an overall picture of the situation. Virtual Context may include the version of the operating system, the interface capabilities, the wireless technology used to accomplish communication, email messages sent and received, and documents edited. Physical context on the other hand may be the presence of another entity, be it a user or device, the proximity to a particular printer, information indicating if the user is standing, walking or sitting or the current weather conditions. When both physical and virtual context are unified a different reasoning about the user's situation may emerge. In the past the main focus was on physical context, with a particular emphasis on location, however there is now a move toward a vast range of possibilities. The Kimura system (Voida et al 2002) accumulates virtual context from the user's desktop using a monitoring system for Microsoft Windows. It manages this using the hooks feature exposed through the Win32 API (Voida 2002). Many systems exist for gathering physical context and especially location, for example the Active Badge System (Want, el al. 1992), the Cricket Location System (Priyantha, Chakrabory, & Balakrishnan 2000), the Radar System (Bahl and Padmanabhan 2000) and the EasyLiving computer vision system (Shafer, Brumitt &Meyers 2000). After the context is gathered and stored in a buffer or repository, a proficient system would merge them and decide what is relevant to the user at the present moment.

3.1.2 Context history

Historical context information, is where computing, user and physical contexts are stored across a time span. Potential uses for this stored information would be to establish patterns of smart space usage. Historical context information is particularly useful for mobile-aware applications. For example applications that predict resource consumption based on observed patterns of mobility and usage. Su et. al. describe a method of mobility prediction in (Su, Lee & Gerla 2002).
Historical context is generally considered to be useful, but it is rarely used in current context-aware systems. Firstly, the context-aware system must decide what historical information is worthy of being kept, and at what level of precision. Evaluating all historical context information that is collected, to acquire the nesscessary information would be prohibitively costly, and efficient algorithms must be implemented to process the wealth of information available to extract meaningful data.

3.2 Characteristics of context information

The characteristics of context are desirable when designing a context-aware system to ensure the context-aware system manages context effectively and efficiently. Researchers at the University of Queensland have classified four characteristics: (Candolin & Kari 2002)

(1) Context Information Exhibits a Range of Temporal Characteristics
Context information has already been subdivided into physical and virtual information; it can be further subdivided into static and dynamic information. Static information is any information related to the user's environment that is invariant. Static information may be retrieved directly from the user. The vast majority of context information is dynamic. Dynamic information must be accumulated continuously, frequently and automatically. Additionally, past context information may be needed to understand the full state of the environment.

(2) Context Information is Imperfect
This considers the validity of context, in particular dynamic context information. Reasons for the doubt over the soundness of the context arise due to the speed at which the context information changes, and the subsequent necessity to process this information before it is used to facilitate work carried out by the user. This "delay between the production and use of context information" (Candolin & Kari 2002) is a concern. Other sources of concern regarding the soundness of context information include the reliability of the producers of the context

information, for example the sensors and the possibility of a broken path between the producers of the context and the point were it is utilised by the user, which means the user may be using out of date information.

(3) Information has many Alternative Representations
There is a considerable difference between the raw data that is gathered from the environment, both virtual and physical, and the processed information that is used to assist the user. The raw data can take on many forms when combined with other context information and when processed by the context-aware system. The probability of the context-aware system obtaining a 100% success rate in "capturing the relationships that exist between the alternative representations" (Candolin & Kari 2002) of the context information and the one apt to the current situation is extremely low.

(4) Context Information is Highly Interrelated
Context information derived from a particular origin may have a very close link with its source, so much so that it is dependant on the origin. Context may not be reliable "where the characteristics of the derived information are intimately linked to the properties of the information it is derived from" (Candolin & Kari 2002).

3.3 Context-Aware System

A context-aware system must be capable of mimicking a human's ability to recognise and exploit implicit information in the environment, in order to advance the operations of its functionalities. Although identifying and deducing a human activity is a challenge, it is critical that context-aware applications should operate by conveying the appropriate information to the right place at the right time through inferring the user's intention. To accomplish this objective the context-aware systems must:

(1) Gather the information from the environment or the user's situation.
(2) Translate this information into the appropriate format.
(3) Combine context information to generate a higher context. A higher context is context information that is derived as a result of the merger of other context information.
(4) Automatically take action based on the retrieved information.
(5) Make the information accessible to the user, immediately, in the future, or when it is required, to enhance and aid in the completion of the user's task.

The researchers of the Context Toolkit at the University of Berkeley propose that there are three categories that a context-aware application can support: (Dey 2000)

(1) Presentation of information and services to a user
(2) Automatic execution of a service for a user
(3) Tagging of context to information to support later retrieval

The Kimura System integrates independent tools into a pervasive computing system. The developers of this system have designed distributed components of a context-aware system. These components fall into three classes (Voida et al 2002), these are:

(1) Context Acquisition, the system gathers context information and adds it to a repository.
(2) Context Interpretation, the system converts the gathered context information into a working context.
(3) User Interaction, the system displays the working context to the user.

3.4 Elements of Context-aware Computing

The steps outlined in section 3.3 ought to be implemented by a context-aware application for an application to advance from a ‘typical’ application into a context-aware application. No model has been standardised for managing context, yet there is a necessity to design a model to optimise the benefits gained from employing context in the application. The management model should handle context in a reusable manner to permit context from one source to be exploited by many distinct applications that perform a variety of tasks.

Context-aware computing is a computing paradigm in which applications can discover and take advantage of contextual information such as location, time of day, people and devices, and user activity. Context-aware computing is especially suited to the areas of mobile and pervasive computing. Instead of adapting systems and applications so that mobility is hidden, context-aware computing provides support for mobile-aware applications.

“*Context-aware computing* is the use of environmental characteristics such as the user’s location, time, identity and activity to inform the computing device so that it may provide information to the user that is relevant to the current context.” (Burrell & Gay 2001)

Context-aware computing was pioneered by XEROX PARC and Olivetti Research Ltd. Context-aware computing involves many individual elements:

• *Sensor technology* - Numerous hardware devices must be equipped with the capability to collect information that will form part of the context of the system. These devices should be relatively inexpensive and readily available, and should hopefully require a minimum amount of configuration and management. They must also be capable of transmitting information to some central location, or else communicating with nearby devices.

• *Context model* - A model of the context information must be formulated to provide a resource for applications to avail of. Projects such as Cameleon (http://giove.cnuce.cnr.it/cameleon.html) are looking at constructing models of context information relevant to applications, thereby “endowing these versions with the ability to dynamically respond to changes in context such as network connectivity, user location and ambient sound and lighting conditions.”

• *Decision systems* - Once the context model has been formed, elements of the system must make decisions based on the information available. These decisions can be made either at application level based on the information available to it, or can be made centrally and then disseminated to the individual agents in the system.

• *Application support* - Application programmers need to be aware of context information, as the addition of this information will fundamentally change how they work. Instead of being driven primarily by explicit user input as they have been in the past, applications will begin to ‘act for themselves’, but must do so in a way that attempts to adhere to the law of least surprise for the user. Ideally, user interaction with a context-aware application should be simpler and more productive than with a traditional application (Cheverst et al 2000).

3.5 Gathering context information

Some context information can be given to the context-aware system explicitly, such as a user's name or age; other context information can be obtained through the use of sensors. Many types of sensor are already commonly in existence and can provide primitive physical information such as light, heat and pressure readings. Other types of context such as facial recognition rely on fairly simple sensors such as cameras, but require considerable processing such as image recognition in order to make use of the information obtained.

Location and identity are the most frequently sensed pieces of context. Active Badges (Want el al 1992) produced by Olivetti and AT&T emit infrared signals which give a rough location and ID. Optical systems for context determination are also possible and research is underway in the areas of optical tracking and motion detection, stereo and 3D reconstruction and object recognition.
Location is an important element of context information. Many different approaches have been taken to determining the location of agents within a context-aware system. GPS, Infrared and radio signals have all been explored. Many context-aware systems that have been produced are *only* aware of location information. While these applications are useful, location is only one element of the wider context information. Schmidt et. al. discuss this fact in (Schmidt, Beigl & Gellersen 1999).

Sensors are not always 100% accurate or reliable, particularly if they are disposable. The information gathering system must be tolerant of sensor failure, and any information gathered from sensors must be subjected to sanity checks to help verify its correctness. Sensor fusion is one method of avoiding this difficulty.

Sensor fusion means aggregating the results of different sensors together to produce a reasonable approximation of the state of the system. This means that some sensors can fail or give erroneous answers, but the system will still be able to determine the real state through the use of a voting mechanism. When considering the output of temperature sensors for example, it might be prudent either to simply average the results or alternatively to discard reported values that differ too greatly from what other sensors report. This method would avoid drastic measures being taken by the context system to correct what it considers to be temperature variations but are actually simply the result of sensor failure. Gellerson, Schmidt & Beigl describe a project making use of sensor fusion in (Gellersen, Schmidt & Beigl 2002).

Sensor discovery is another issue: with the proliferation of sensors that will be required for a context system to be useful, configuring and managing a large number of sensors can be a difficult task. It is possible (perhaps even likely) that there would be more sensors than people in such a system, so being able to automate the detection and configuration of sensors would be necessary. The MUSE project (Castro & Muntz) uses Jini for automatic sensor detection.

3.6 Retrieving Context

Push and pull are the two options available to a context-aware system to extract the necessary context information from context sources. Context information is periodically sent to the application in the push model. This means that the context source collects the context information before it is needed, which may result in a better performance. The shortcoming of this approach is the consumption of resources for gathering and disseminating context that may never be exploited by the context service. The pull model in contrast gathers only the context information that is required by the service, however this approach exposes the context service to network delays and unavailability. In the case of the push model, a trade off between freshness of context information with the cost of frequent updates must be measured (Ebling, Hunt & Lei 2001), while the pull approach must consider the trade off between reserving resource and the probability of faults.

3.7 Models for Managing Context

The first mandatory step in managing context is to supply the context-aware applications with a software component that offers access to context information. Three models that present a structured approach to providing context to an application are:

- Context Widgets
- Infrastructure approach
- Blackboard

3.7.1 Context Widgets

The context widget, which is the most prevalent model, resides between the context-aware application and the environment, comparable to the way in which the GUI widget resides between the application and the user. The main objective of the widget is to separate the application from context acquisition issues. This separation hides the complexity of gathering and managing context information, thus the approach and problems related to accumulating the context has trivial or perhaps no impact on the application. In addition the widget is used as a mediator to pass only the pertinent information to the application, any other accumulated context information does not concern the application. Finally the context widget provides a reusable building block, which permits many applications to exploit context information that is detected by the sensors. A context widget functions independently of applications, which permits multiple applications to use it simultaneously. A context widget is also responsible for maintaining a complete history of the context acquired for the user's situation.

3.7.2 Infrastructure Model

This model is equivalent to the client-server model (Martinka 1996). It is a more flexible than the context widget, as it promotes the independence of the components in the context-aware system. Each component is empowered with the capability to perform all the required functionality, including establishing connections, organising input and output messages and managing faults, significantly increasing this model's complexity. This model supports a greater range of devices and applications by using standard coding and networking protocols. It also facilitates both independent and dynamic changes in the different components, be it the sensors, services or devices. Finally the third advantage of this model is the straightforward approach available for developing and deploying sensors, processing power, data and applications. (Hong & Landay 2001)

3.7.3 Blackboard

The blackboard model adopts a data-centric point of view. When applying the blackboard model the application "posts messages to a common shared message board, and can subscribe to receive messages matching a specified pattern that has been posted." (Dey & Abowd 2000) Many problems exist when all communication must traverse a central database (the common shared message board), for example the existence of a single point of failure.

3.7.4 Trade-off criteria

When deciding on the most apt model for a context-aware system it is vital to consider trade-off characteristics. The different dimensions that should be measured are efficiency, effort in configuring, robustness, simplicity and extensibility. (Winograd 2001)

Efficiency

The efficiency of the model depends on the bandwidth and latency. The objective of the model, in terms of efficiency, is to accelerate the throughput of information. It could be argued that efficiency should not be an issue with the present networking and processing speed, but when the explosion in the number of networked applications and devices are considered, it becomes apparent that efficiency is still a major concern.

Effort in Configuring

These context-aware systems comprise many components, so it is imperative that adding, removing or changing component is not a tedious task. Configuring individual components of the system must be achievable without the disruption or total failure of the system when possible.

Robustness

The context-aware system should be designed to continue to operate in a failure mode when components fail. When a failure mode is not possible, the system should fail in a controlled manner. The degree to which the system can cope with failure is termed its robustness.

Simplicity

It is important that the model is not excessively complex because "a system that requires complex understanding by system builders in order to make use of its facilities will be used only by those who have the dedication and motivation to master it." (Winograd 2001)

Extensibility

The concept of context-aware computing is still looked upon as an emerging technology thus it is important that "services supporting a general notion of context must be easily extensible to accommodate new and unanticipated sources of context information." (Ebling, Hunt & Lei 2001)

3.7.5 Comparisons

The trade-offs described in section 3.7.4 can be used to compare and contrast the different models for context management. The widget model has tight coupling of system components, which may make it the most efficient model in some circumstances, however it suffers from complex configuration and issues with powerlessness in the cause of failure. The infrastructure model with its independent components may be extremely complex, which is a no factor for many developers, however its positive criteria of straightforward configuration and robustness may compensate for the negative implication of complexity. Finally the blackboard model with its very loosely coupled components may suffer from efficiency problems, but it is a simple, robust, easily configurable model. (Winograd 2001)

Although the models assist in managing context, further abstractions are essential to handle context information effectively. These abstractions are detailed in section 3.8 to 3.10 inclusive. .

3.8 Interpreters

After sensing, acquiring and saving the context from various sources the next activity for the context-aware system is to produce a mechanism for achieving context interpretation, so that the gathered context information is utilised in a fitting manner. "Interpretation refers to the process of raising the level of abstraction of a piece of context." (Dey & Abowd 2000) The interpretation may involve integrating numerous contexts into one to provide a higher-level context, thus the interpreter alters context information by raising its level of abstraction.

One approach is to use context fusion to convert the "lower level context into higher level usable by applications". (Mitchell 2002) Context fusion may be viewed in different ways. One could regard it to be the synthesis of context from the same type of sources in order to increase the validity of the information so that erroneous sensors or reading are detected to avoid improper decisions by the system. Context fusion is also deemed to be the aggregation of context of varying types from a different variety of sources to produce a context that is exploitable by the system.

The Technology for Enabling Awareness (TEA) project has produced a layered architecture for fusion. The layers include Sensors, Cues, Context, and Scripting. The Sensor layer has physical sensors for measuring physical factors in the environment and logical sensors for gathering host criteria. The Cues layer produces an abstraction from the physical and logical sensors. The Context layer derives a depiction of the situation from the abstractions in the Cues layer. Finally the Scripting layer "provides a mechanism to include context information in application" (Schmidt, Beigl & Gellersen 1998) The Scripting layer has three states: (1) entering a context, if a situation is specified with a probability that is higher than a threshold an action is performed after a certain time. (2) Leaving a context, if a situation is specified with a probability that is less than a threshold an action is performed after a certain time. (3) While in a context, if a situation is specified with a probability that is higher than a threshold an action is performed every specified time interval. (Schmidt, Beigl & Gellersen 1998) "Context fusion must handle seamlessly handing off sensing responsibility between boundaries of different context services. Negotiation and resolution strategies need to integrate information from competing context services when the same piece of context is concurrently provided by more than one service." (Abowd & Mynatt 1999)

3.9 Aggregators

"Aggregation refers to collecting multiple pieces of context information that are logically related into a common repository." (Dey & Abowd 2000) This abolishes the need for the context-aware application to gather the required context from different sources that would otherwise be obligatory as a consequence of the distributed nature of the context-aware systems. Aggregators support the delivery of particular context to an application, by accumulating related context that the application seeks into one logical placement. An aggregator facilitates interpretation of context hence it will aggregate diverse context information for different requesting applications. Individual applications can be alerted to alterations in the aggregator's context and can push or pull context from the aggregators. "Aggregators provide an additional separation of concerns between how context is acquired and how it is used." (Dey & Abowd 2000)

3.10 Context Services

The three management models outlined along with the two abstractions, interpretations and aggregators, described above deal primarily with the acquisition and manipulation of context information to provide the application with relevant information in an apt manner. If these components are implemented by a context-aware system three of the five steps, outlined in section 3.3, necessary in developing a proficient system will be realised.

The context-aware system will automatically take action based on the retrieved information. "Services are components …that execute actions on behalf of applications," (Dey & Abowd 2000) they use actuators to manipulate or change the state of the environment in response to a decision made by the application based on acquired context. Context service may be synchronous or asynchronous. An asynchronous context service "requires that the application waits for a response" (Ebling, Hunt & Lei 2001) but because there is no guarantee that a response will result, the application requesting the response will be notified that the service has been initiated and the service results will be delivered when they are available. (Dey & Abowd 2000) A synchronous context service requests that the application waits for a response.

3.11 Application support

Chen and Kotz (Chen & Kotz 2000) divide the awareness that applications have of context information into two different types:

• *Active Context Awareness* - an application automatically adapts to discovered context, by changing its behavior.

• *Passive Context Awareness* - an application presents the new or updated context to an interested user or system. Alternatively the application makes the context persistent for the user or system to retrieve later.

Once context information has been obtained, the use of passive context information can be implemented in an application. Simply displaying the data from a sensor or the current location of a person would be use of passive context. The case of active awareness of context is the more challenging, as it requires fundamental changes in the way applications operate. While the user is somewhat 'in control' of the current generation of applications (requiring the user to initiate almost all action), there is the danger that users could start to feel helpless and out of control of systems and applications that begin to make decisions on their own because they have more information than the user does.

With context-aware systems, applications can exist that can adapt and need less explicit input from the user. For example, many devices could use context information to do things such as turning on and off depending on whether they are (or are likely to be) used, using a discreet mode depending on situation, sharing a controlled amount of information with other systems with the user's authorisation, adapting information output based on terminal types, etc. Applications must provide support for these capabilities, but if these systems make decisions based on the information available to them, the user will perceive a better result and be more productive.

3.12 Discoverers

Another useful component for a proficient context-aware system is a discoverer. A discoverer may be labelled the active management component, as it supervises the activities of the other components in the context-aware system. The discoverer maintains a registry of the components and capabilities of the system, thus if a component is added, removed or modified the discoverer's registry is updated. If any of the system components fail, the discoverer updates its register indicating that the particular component is no longer available for use. The main purpose of the discoverer is to supply the other components in the system with data relating to one another, thus it is vital that the discoverer's registry is updated promptly. (Dey & Abowd 2000)

3.13 Privacy

Personal user information is a necessity to construct a system, which adapts according to a user's needs and goals, but this raises the issue of user privacy. Context-aware systems raise privacy challenges not previously considered by traditional systems. "Privacy is intrinsically bound up with control" (Weitzner, Ackerman, Darrell 2001) so control of the system must lie in the user's hands if complete user privacy is to be realised. However in most cases this is not viable, as the user will not be allowed to participate in control or because a user will not want to administer the control when acquisition and dissemination of context information is not simple. All the information that is useful to context-aware applications must be considered carefully in each situation that it is used, to avoid giving applications too much information which could be a risk to users' privacy. A method of information spaces for ensuring users' privacy and protection of data is described by Jiang and Landay in (Jiang & Landay 2002), and also by Bellotti and Sellen (Bellotti and Sellen 1993). A balance must be struck between the advantages of disclosing personal information (enabling the application to tailor services to users' needs) and the disadvantages of such disclosure in the loss of privacy.

Privacy rules may state that:

(1) A user should be notified about the type of information collected,
(2) A user should have the choice to halt the collection of personal information,
(3) A user should have the ability to access the information that is gathered and delete information if desired.
(4) Security to eliminate the possibility of unauthorised persons accessing the user information should be in place.

It is not clear how privacy rules will be governed, as different people will desire different privacy rules. In addition Human Computer Interface (HCI) problem will have to be resolved, complex information about a privacy policy will have to be displayed to the user in a manner that is easily understandable, so that the user will take the necessary steps to protect personal information without being diverted from of the main task at hand. Context-aware computing introduces extra privacy concerns but it may also be applied to "aid the negotiation over privacy agreements… by considering what privacy choice the user made the last time she entered this particular context, what privacy choices have the user's trusted colleagues made in the current context, and what privacy choices did the user make when dealing with the same data collector though in an otherwise different context." (Weitzner, Ackerman, Darrell 2001) The MIT Oxygen project is currently developing a context prototype with user privacy as the foremost driving factor.

3.14 What context lets us do

There are many sample uses for context-aware systems, in fact the applications of such systems are mostly limited only by one's imagination. Applications that have been suggested so far can be roughly broken down into the following categories, based on those described in (Schilit, Adams & Want 1994):

• *Context-triggered Action* - Simple if-then rules that specify how context-aware systems should adapt. For example, if the user is in a meeting and there is a phonecall for him or her, then the call should be routed to voicemail. Triggers can also be placed on certain events, to make the context system take action such as notifying an administrator of an increase in temperature in a server room.

• *Contextual Information and Commands* - commands issued by the user can produce different results depending upon the context in which they were issued.

• *Proximate selection* - given information about the location of the user and the situation of the user, objects located nearby can be emphasized more or made easier to choose in the user interface. This prioritisation would be particularly useful on a mobile device which might have limited screen real estate. For example, a nearby printer would be proposed before one further away when the user wishes to print something. The same idea would hold true for any location-based service the user would wish to avail of.

• *Automatic Contextual Reconfiguration* - adding components, deleting components, or changing the connections between components based on current context. The issues that arise have synergies with the field of service composition (Kiciman, Melloul & Fox 2001). For example, Schilit et. al. produced a system whereby anyone carrying a PARCTAB (Schilit et al 1993) and entering a room would have it bound to an instance of a virtual whiteboard associated with that room. When the user was finished with the whiteboard and left the room, the PARCTAB’s connection to the whiteboard was broken. In this situation, the context information on the user’s location was used to configure the connection between the mobile device and the ‘static’ virtual whiteboard.

• *Metadata Tagging* - Context information can be attached to existing pieces of information, to give us more implicit information about objects in our system. This context acts as metadata about both physical and virtual objects in our system. For example, when a user records an audio clip on a handheld device, the system can attach the current context information (date and time, people present, current activity) to the clip for easy retrieval and indexing. Diaries can be created automatically, simply by ‘recording’ relevant context.

• *Terminal adaptivity* - the usability of mobile devices suffers from small and cluttered user interfaces. Context information can help to present the user with only relevant information, rather than overwhelming them with many different options.

The most frequently created context-aware application makes use of the user’s location information by acting as a sort of virtual tour guide such as Cyberguide (Abowd et. al 1997) from Georgia Tech, or the Teleporting System (Bennett, Richardson & Harter 1994) from Olivetti Research Ltd. Such a system uses GPS or infrared tracking to determine user location with a reasonable degree of accuracy. It displays this location on the screen of a handheld device, and uses that knowledge to display information on the screen when the user is near to a point of interest. It can also be used as a log, to record places that have been visited by the user for later examination. Simple applications of such a location system would be to redirect computer output to a nearby screen, or to automatically forward a user’s phone calls to the nearest phone (Want et al 1992).

PDAs which have sensors attached are a low-cost entry to a context-aware system. By fitting PDAs with sensors such as proximity sensors, touch sensors, orientation sensors, light sensors and temperature sensors, the device itself can adapt to this context information. For instance, the user can choose whether to use the device in portrait or landscape mode by simply rotating the screen. Power management can be made more intelligent by telling the device to turn on if being held and tilted, and turn off if left idle with nobody nearby. These self-contained context-aware devices are attractive because they can make use of context without the need for the surrounding environment to be instrumented.

3.15 Examples of Context-Aware Systems

Four examples of system that use context information are described in this section. These examples show different approaches to managing context and display the vast amount of areas and issues that have to be addressed when using context in a system.

Context Toolkit
The context toolkit supports context-aware applications by assisting non-context-aware applications to use context and by evolving existing context-aware applications. It shields the applications from changes in the context and the consequence of the changes. The context toolkit separates the application from context acquisition issues through the use of widgets, it also stores context gathered from the environment and constructs a history, which is made accessible to applications to make decisions on the intent of a user. The context toolkit encompasses a context interpreter to perform functionalities as outlined in the interpreter section and an aggregator to accumulate related context that the application seeks into one logical placement. (Dey 2000)

Owl
A context-aware system currently under construction by Ebling et al is the Owl context service, which "aims to gather, maintain and supply context information to clients. It tackles various advanced issues, including access rights, historical context, quality, extensibility and scalability." (Candolin & Kari 2002) "It offers a programming model that allows for both synchronous queries and asynchronous event notifications. It protects people's privacy through the use of a role-based access control (RBAC) mechanism." (Ebling, Hunt & Lei 2001)

Kimura
The motivation of the Kimura System is to integrate both physical and virtual context information to enrich activities of knowledge workers. It utilises a blackboard model based on tuple spaces. The four components that operate on the tuple spaces are: (MacIntyre, Mynatt & Darrell 2001)

(1) Desktop monitoring and handling components, which uses low-level Window hooks to observe user activities and the interpreter component
(2) The peripheral display and interaction components that read and display the context information so that the user can observe and utilise the context in tasks.
(3) Context monitoring component that writes low-level tuples, which are later interpreted.
(4) Interpreter component, which translates low-level tuples into tuples that can immediately be read by the whiteboard display and interaction component.

Solar
The Solar system is a middleware system designed to support context-aware applications. The system comprises of information sources, which are sensors that gather physical or virtual context information,

filters, transformers and aggregators to modify context to offer the application practical context information. Researchers at Dartmouth College are designing and experimenting with this system, aspiring to produce a system that is flexible, scalable and reusable.

4. Research Directions

Location was unquestionably the initiator of context-aware computing, however from this paper it should be clear that although context-aware computing is still a relatively new area it has advanced far beyond the premise of location. The chief goal is to utilise as many types of context information as possible in a manner that enhances the functionality of the application and abolishes much of the effort contributed by the user. At present current mobile applications are being designed with the same location-independent mindset as traditional applications. The advantages of context information for mobile applications seems clear - applications of all types will be able to avail of a much richer set of information with which to make decisions. However, providing this information to applications in a timely and efficient manner is a challenge. High mobility applications need to be aware of changing context, even more so than their low-mobility counterparts.

Research must be carried out into the modelling of users in context systems, and into the privacy of those users. User modelling is a mature field in other areas, but there is a definite need for the expertise to be applied to context systems, and privacy is such an important issue that it must be borne in mind in all stages of the design of such systems.

The design of context-aware systems must facilitate the prevention of potentially embarrassing or dangerous situations for the end user. There are situations in which the forwarding of phone calls could be inconvenient or even dangerous, for example if a surgeon were conducting an operation. Context-aware systems need to conduct 'risk management' in order to take into account the costliness of mistakes, and factor that information into the decisions they make.

A context-aware system needs to determine how much value context will add to each individual situation, and not rely on it as a panacea. At least in the short term, context information could be unwieldy and difficult to make use of. It seems clear though that there is enough value in the potential of context-aware computing to persevere.

Strides in sensors, recognition, and wireless networking are enabling new classes of context-aware applications. It is however still necessary to lower the barriers to entry of these systems. Current methods of context provision are extremely application-specific and non-general. Context-aware systems will only become widely used once there are standard mechanisms for the sharing of context. There are many technical issues in this field, and even more people-related issues. When these issues are overcome however, there is much potential here for new kinds of interaction and applications that make use of context.

A standard is required for context-aware computing to eliminate repetition of effort experienced by developer. This standard must be put in place to allow reuse of context-aware system components so that an entirely new system need not be produced for every context-aware application. If this context standard is not fabricated the benefits of context may never be realised as the production of context-aware systems will never become prevalent. Another aspect that most be addressed in the near future is the security and privacy problems produced by context-aware computing. Mechanisms to guarantee security of personal information must be studied to give users of context-aware applications easy of mind. Some models and abstractions that may be considered in the production of a context standard and problems that need to be addressed and resolved for privacy have been outlined in this paper.

This survey of work in the area of context-aware computing is enlightening in the different approaches to classifying types of context information. It seems clear that almost any type of information could be regarded as context, so it is important to classify this information into different types in order to be able to use it efficiently, as has been addressed by many researchers in the field. This classification will assist us in managing context information, by being aware of its individual characteristics. One can however never hope to enumerate all the information that could constitute context.

Research is also being carried out into the concept of modelling of users, systems and devices for context-aware systems. This research will hopefully lead to a reusable framework for the storage of context information, which will enable different context systems to communicate with each other and share context. Only once context information is truly ubiquitous will its full potential be realised. However it is possible to build up from simple applications of context to a more advanced level, which fully leverages the power of context-aware computing.

Candidate management systems for context are being explored and will hopefully result in a system that is flexible and intuitive, both for the user and manager of a context system. There are many issues still to be addressed, however the work so far explored leads us to believe that the future for context-aware computing is bright.

5. References

Abowd Gregory, Ebling Maria, Hunt Guerney, Lei Hui & Gellersen Hans-Werner, (2002), Context-Aware Computing, *IEEE Pervasive Computing,* IEEE

Abowd Greogory D. and Mynatt Elizabeth D. (1999), Ubiquitous Computing: Past, Present and Future, Available: http://www.cc.gatech.edu/classes/AY2000/cs7470_spring/readings/tochi.PDF [2002, Nov. 27]

Albrecht Schmidt, Michael Beigl, and Hans-W. Gellersen (1999). There is more to context than location. *Computers and Graphics*, 23(6):893–901.

Bill N. Schilit, Norman Adams, Rich Gold, Michael M. Tso, and Roy Want (1993). The PARCTAB mobile computing system. In *Workshop on Workstation Operating Systems*, pages 34–39.

Bill Schilit, Norman Adams, and Roy Want, (1994). Context-aware computing applications. In *IEEE* Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, US.

Cameleon project - context aware modelling for enabling and leveraging effective interaction; http://giove.cnuce.cnr.it/cameleon.html.

Candolin Catharina and Kari Hannu H. (2002), Modelling Context Information in Pervasive Computing Systems, Available: http://www.ietf.org/internet-drafts/draft-candolin-cam-00.txt [2002, Nov. 27]

Dey Anind K. and Abowd Greogory D. (2000), A Conceptual Framework and Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications,
Available: http://www.cc.gatech.edu/fce/ctk/pubs/HCIJ16.pdf [2002, Nov. 27]

Dey Anind K. (2000),Understanding and Using Context,
Available: http://www.cc.gatech.edu/fce/ctk/pubs/PeTe5-1.pdf [2002, Nov. 27]

Ebling Maria R., Hunt Guerney D. H. & Lei Hui (2001), Issues for Context Services in Pervasive Computing, Available: http://www.cs.arizona.edu/mmc/13%20Ebling.pdf [2002, Nov. 27]

Emre Kiciman, Laurence Melloul, and Armando Fox, (2001). Towards zero-code service composition. In *Eighth Workshop on Hot Topics in Operating Systems*, Elmau, Germany, May 2001.

Frazer Bennett, Tristan Richardson, and Andy Harter, (1994). Teleporting - making applications mobile. In *IEEE Workshop on Mobile Computing Systems and Applications*, pages 82–84, Santa Cruz, California, December.

Gregory D. Abowd, Anind K. Dey, Robert Orr, and Jason A. Brotherton (1997). Context-awareness in wearable and ubiquitous computing. In *ISWC*, pages 179–180.

Gregory D. Abowd, Christopher G. Atkeson, Jason Hong, Sue Long, Rob Kooper, and Mike Pinkerton (1997). Cyberguide: A mobile context-aware tour guide. *ACM Wireless Networks*, 3:421–433.

Guanling Chen and David Kotz, (2000). A survey of context-aware mobile computing research. Technical report, Department of Computer Science, Dartmouth College, 2000.

Hans-Werner Gellersen, Albrecht Schmidt, and Michael Beigl (2002). Multi-sensor context-awareness in mobile devices and small artefacts. *Journal on Mobile Networks and Applications, Special Issue on Mobility of Systems, Users, Data and Computing in Mobile Networks and Applications*, October 2002.

Hong Jason I. and Landay James A. (2001), An Infrastructure Approach to Context-Aware Computing, Available: http://guir.berkeley.edu/projects/cfabric/pubs/context-essay-final.pdf [2002, Nov. 27]

Jenna Burrell and Geri K. Gay (2001). Collectively defining context in a mobile, networked computing environment. Short talk summary in CHI 2001 Extended abstracts, May 2001.

Joseph J. Martinka 1996, Requirement for Client/Server Performance Modeling Available : http://www.hpl.hp.com/techreports/95/HPL-95-96.pdf [2003 Jan 16]

Keith Cheverst, Nigel Davies, Keith Mitchell, and Christos Efstratiou (2000). Using context as a crystal ball: Rewards and pitfalls. In *Proceedings of Workshop on 'Situated Interaction in Ubiquitous Computing' CHI*, 2000.

MacIntyre Blair, Mynatt Elizabeth D., Tullio Joe & Voida Steve (2001), Hypermedia in Kimura System, Available: www.cc.gatech.edu/fce/ecl/projects/kimura/pubs/kimura-hypertext2001.pdf [2002, Nov. 27]

Mitchell Keith (2002), A Survey of Context-Awareness, Available: http://www.comp.lancs.ac.uk/~km/papers/ContextAwarenessSurvey.pdf [2002, Nov. 27]

Morse David R., Armstrong Stephen, Dey Anind K. (2000), The What, Who, Where, When and How of Context-Awareness, Available: http://www.cc.gatech.edu/fce/ctk/pubs/PeTe5-1.pdf [2002, Nov. 27]

Nissanka B. Priyantha, Anit Chakraborty, & Hari Balakrishnan (2000), The Cricket Location-Support System Available: http://nms.lcs.mit.edu/papers/cricket.pdf [2003 Jan 16], Proc. of the Sixth Annual ACM International Conference on Mobile Computing and Networking (MOBICOM)

Paramvir Bahl and Venkata N. Padmanabhan (2000) RADAR: An In-Building RF-based User Location and Tracking System Available: http://www-bsac.eecs.berkeley.edu/~ldoherty/radar.pdf [2003 Jan 16] Microsoft Research
Paul Castro and Richard Muntz (2000). Managing context for smart spaces. *IEEE Personal Communications*, October 2000.

Roy Want, Andy Hopper, Veronica Falco, Jonathan Gibbons (1992), The Active Badge Location System Available at:
http://citeseer.nj.nec.com/cache/papers/cs/13941/http:zSzzSzwww.parc.xerox.comzSzcslzSzmembersz SzwantzSzpaperszSzab-tois-jan92.pdf/want92active.pdf [2003 Jan 16]

Roy Want, Andy Hopper, Veronica Falcao, and Jon Gibbons, (1992). The active badge location system. *ACM Transactions on Information Systems*.

Schmidt Albrecht, Beigl Michael, Gellersen Hans-W. (1998), There is more to Context than Location,Available:
http://www.comp.lancs.ac.uk/~albrecht/pubs/pdf/schmidt_cug_elsevier_12-1999-context-is-more-than-location.pdf [2002, Nov. 27]

Schilit Bill N., Adams Norman & Want Roy (2002), Context-Aware Computing Applications, Available: http://www.fxpal.com/people/archive/schilit/wmc-94-schilit.pdf [2002, Nov. 27], IEEE Workshop on Mobile Computing Systems and Applications, IEEE

Shafer, S., Brumitt, B., and Meyers, B. (2000), The EasyLiving Intelligent Environment System, CHI Workshop on Research Directions in Situated Computing Available:

Http://www.research.microsoft.com/easyliving/Documents/2000%2004%20Steve%20Shafer%20CHI.doc [2003 Jan 16]

Victoria Bellotti and Abigail Sellen (1993). Design for Privacy in Ubiquitous Computing Environments. In *Proceedings of the Third European Conference on Computer Supported Cooperative Work*, pages 77–92. Kluwer.

Voida Stephen, Mynatt Elizabeth D., MacIntyre Blair & Corso Gregory M. (2002), Integrating Virtual and Physical Context to Support Knowledge Workers, *IEEE Pervasive Computing,* IEEE

Mark Weiser (1993), Ubiquitous Computing, IEEE Computer "Hot Topics"

Weitzner Daniel J., Ackerman Mark & Darrell Trevor (2001), Privacy In Context Available: http://www1.ics.uci.edu/~jpd/NonTradUI/SpecialIssue/ackerman.pdf [2002, Nov. 27]

W. Su, S. Lee, and M. Gerla (2000). Mobility prediction in wireless networks.

Winograd Terry (2001), Architectures for Context, Available: http://hci.stanford.edu/~winograd/papers/context/context.pdf [2002, Nov. 27]
HCI Journal, *Lawrence Erlbaum Associates, Inc, Mahwah, NJ*

Xiaodong Jiang and James Landay (2002). Modeling privacy control in context-aware systems using decentralized information spaces. *IEEE Pervasive Computing*, 1(3), July-September 2002.