2. Background and Literature Review

The literature review presents a discussion and critical review of the research relating to context awareness and constructive memory. It provides an overview of the theoretical background of the research disciplines, identifying the approaches adopted, and discusses the benefits and challenges posed. It also highlights the principal areas in which solutions fall short of requirements, motivating the research presented in the chapters to follow. Section 2.1 describes the concepts of context and analyses the context-aware lifecycle processes. Section 2.2 reviews the methods of constructive memory and reasoning. Section 2.3 summarises the reviewed work, comparing them collectively to the research aims and objectives.

2.1 Context Awareness

At the core of ubiquitous computing is *context awareness*, the concept of sensing and reacting to dynamic environments and activities. Schilit et al. (1994) first introduced the term context awareness. It has grown into a technology which is incorporated in a variety of applications as a means to facilitate higher levels of interaction between users and applications. Context awareness emerged from the field of pervasive or ubiquitous computing as a technique for instilling applications with an awareness of their surroundings, in order to achieve a high degree of autonomy and flexibility. The chief goal is to reduce the burden on users when interacting frequently with and driving their applications, by creating software that is un-intrusive and largely invisible. This design goal is often termed invisible computing (Norman, 1998) and is becoming more relevant as computing spreads to environments in which user attention is at a premium, as the ratio of applications to users grows.

Despite the appearance of a spate of context-aware applications in recent years, the concept of context remains ill-defined. Dey (2001) defined context as “any information that can be used to characterise the situations 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 the applications themselves” (Dey, 2001, p. 5). Dey et al. (1999a) also described context as the implicit situational information which humans use when talking with humans.

Context has been variously characterised as an applications environment or situation (Brown, 1995; Hull et al., 1997) and as a combination of features of the execution environment, including computing, user and physical features (Schilit et al., 1994). Schilit et al. (1994) conducted research which advanced context-aware computing into popularity and define context-aware computing as software that adapts according to its location of use, the collection of nearby people

and objects, as well as changes to those objects over time. According to Schilit et al. (2002), there are three processes in the context-aware lifecycle:

- context discovery gathers and captures the contextual information;
- context interpretation of the captured information and the selection of useful contextual information; and
- context abstraction and use involves the application utilising the contextual information and responding appropriately.

This thesis adopts these three processes and will use them to frame the research in describing the context-aware lifecycle.

In addition to being able to obtain the contextual information, it must include some way of processing this information to be meaningful to the user. This is probably the most challenging problem. According to Abowd et al. (1999), this is due to the richness of the language that humans share, the common understanding of how the world works, and an implicit understanding of everyday situations, which is enabled by usage of implicit situational information, that is context. Unfortunately, when humans interact with computers, the ability to convey contextual information does not transfer well. There is a need for computers to tailor the provision of contextual information more closely to needs of the user to improve on this communication in human-computer interaction.

Location is a crucial part of the context awareness mechanism for ubiquitous computing (Mantoro and Johnson, 2003), and much research in the past has focused on location-sensing technologies, location-aware application support and location-based applications. The awareness of location comes from one person being able to textually or verbally describe a location to another (Dearman, 2004). This is problematic, as communication of instructions or descriptions between persons may be ambiguous, misinterpreted or misunderstood. People may not even know their current location. Location-aware computing avoids the complications associated with verbal and textual communication by providing visual cues and references (Dearman, 2004).

Let us observe that location and context are not completely disjointed concepts. In fact, location (and the distance between clients and their needed resources) is often a primary factor in determining the applicable context (Dey, 2001; Leroy et al., 2003). However, the distinction between location and context is relevant. For example, two mobile users/devices may be at the same location but perceive different contexts because of their different preferences/capabilities and their belonging to different administrative domains. Moreover, the distinction can facilitate service customisation. Depending on application-specific requirements, service designers can

decide to exploit the visibility of either the location or the context, or both, in order to tailor the service results accordingly (Bellavista et al., 2006).

Bellavista et al. (2006) provided an intuitive example of the need for location/context-dependent services in the case of a museum guide assistant. The location information permits immediate access to the descriptions of only the artworks in the currently visited museum room. Context awareness permits user profiles to be specified in order to adapt the delivery and presentation of data according to user preferences. Bellavista (2006) defines:

- context as a group of available resources and services whose grouping property is based on whichever logical relationship is considered relevant for the supported service provisioning; and
- location as the property of physical position of users, devices and resources.

2.1.1 Context Discovery and Sensor Technologies

Context discovery is the first phase in the context-aware lifecycle. Context discovery is concerned with capturing all of the contextual information which is available from the user in a context. Specifically, determining the location of people and objects has been the focus of much research in context awareness and ubiquitous computing. Context is defined as where the user is, who they are with and what they are doing. All of this contextual information can inform the mobile computing device for future interaction with an application. Ongoing research is concerned with discovering the most effective approach to capture contextual information. The derivations of useful, high-level contextual information from sensors are some of the most fundamental problems in the development of context-aware applications. The focus has shifted gradually from the abstraction of information from a simple, uniform sensing infrastructure to more general techniques for extracting rich types of information from diverse collections of sensors and technologies.

A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task (Jiang and Steenkiste, 2002). The main problem with context adaptation is that the context cannot be easily identified and measured. The location of the user is an element of the context that currently can be measured more or less accurately, depending on the positioning system in use.

There are well established context discovery technologies for position location, such as GPS, mobile phone triangulation, and technologies such as digital TV and WiFi. Borriello and Deshpande (2002) classified these technologies into two categories: network-based or device-

based. Location sensor technologies can also be classified in terms of the environment suited for them, such as indoors vs. outdoors positioning. These technologies have their own merits and drawbacks. Some work best outdoors and some work best indoors. The performance variation of these technologies with regard to the environment has been the main reason for their development and continued existence. Along with these technologies, there are emerging mechanisms for position location such as digital TV signals, ultra-wideband signals or WiFi signals (Borriello and Deshpande, 2002).

Different context-aware environments require different location sensor technologies, partly because there are various possible environments where platforms have to determine position location. Since each technology is specifically targeted for a specific environment, there is a mix of technologies that must be supported by a single platform that moves seamlessly between these environments (Borriello and Deshpande, 2002). Evidence of this can be demonstrated from a scenario whereby one is trying to locate a user inside a building; the platform may need to depend on a technology that is suited for indoor environments, such as WLAN. On the other hand, when the same user walks out of the building and requests directions to a point of interest, the platform may use GPS technologies.

For context-aware applications in open and outdoor areas, GPS is a common choice. A GPS receiver estimates position by measuring satellite signals' time difference of arrival. Although GPS offers near global coverage, its performance degrades indoors and in high-rise urban areas, and receivers have relatively long start-up time and high cost (Hazas et al., 2004). Many of the location-sensing systems developed since then are based on radio signals and frequencies (Balakrishnan and Priyantha, 2003). By using base station visibility and signal strength, it is possible to locate WiFi-enabled devices with accuracies from several metres to tens of metres. Bluetooth technology, which offers a shorter range then WiFi, can provide more accurate positioning but at the expense of requiring more fixed-based stations to provide coverage. Inexpensive RFID tags can be used for location determination as well, by placing RFID readers in doorways and other strategic points to detect the passage of people or objects. Ultrasound can be used to determine distances between mobile tags and known points in the environment. A process akin to triangulation can then be employed to derive a location estimate for the tag. One type of ultrasonic ranging device is Cricket, an indoor location system developed at the Massachusetts Institute of Technology (MIT) (Balakrishnan and Priyantha, 2003). The MIT Cricket project is explained in more detail in Section 2.1.4.

Hazas et al. (2004) illustrated (Figure 2.1) the current and predicted deployment of context discovery and sensor technologies, in respect to location, within the next four to five years for context-aware applications. The widest existing deployments are based on GPS, which is particularly suited to outdoor applications. Other current deployments are found in vertically integrated solutions and comprise a specific context-aware application, appropriate location sensing hardware and a custom software platform. Looking ahead, numerous factors are accelerating the adoption of coarse grained context discovery technologies (Hazas et al., 2004). WiFi, Bluetooth and other wireless networking technologies have led to many end-user devices being equipped with such hardware that can be used for location in context-aware applications.

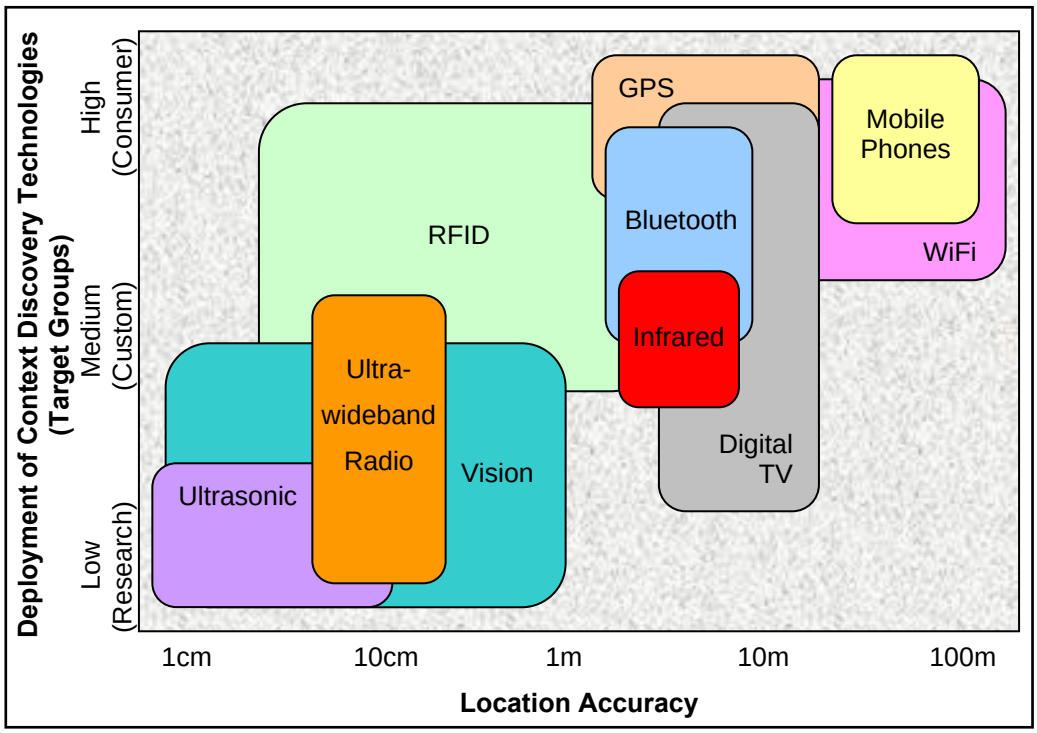

Figure 2.1. Context discovery technologies with each box's horizontal span displaying the range of accuracies that technology covers. The bottom boundary represents the current deployment, while the top boundary shows the predicted deployment over the next four to five years (Hazas et al., 2004).

2.1.2 Context-aware Models and Interpretation

Context interpretation is the second phase in the context-aware lifecycle. The context interpretation process is responsible for selecting and transforming the contextual information gathered in the context capture process into a piece of information that is useful. The main components of context-aware computing and applications are sensors for perceiving the context data, a set of rules governing behaviour according to the context and a set of actuators for

generating responses. Context interpretation involves integration of various types and sources of discovered contextual information into one useful entity. As with the context discovery and capture, there have been quite a few notified attempts to perform the context interpretation process effectively.

Biegel and Cahill (2004) designed the sentient object model (Figure 2.2) for context-aware applications in mobile environments. The sentient object model is an encapsulated entity with its interfaces being sensors and actuators. The sensors produce software events in reaction to real world stimuli, whilst actuators consume software events and react by attempting to change the state of the real world via a mobile device (Biegel and Cahill, 2004). The appeal of this model is its organised approach to developing context-aware applications supporting the aspects of sensor fusion, context extraction and context reasoning. In terms of developing the computational model, it is concerned with the approach to reasoning-based on a hierarchy of contexts.

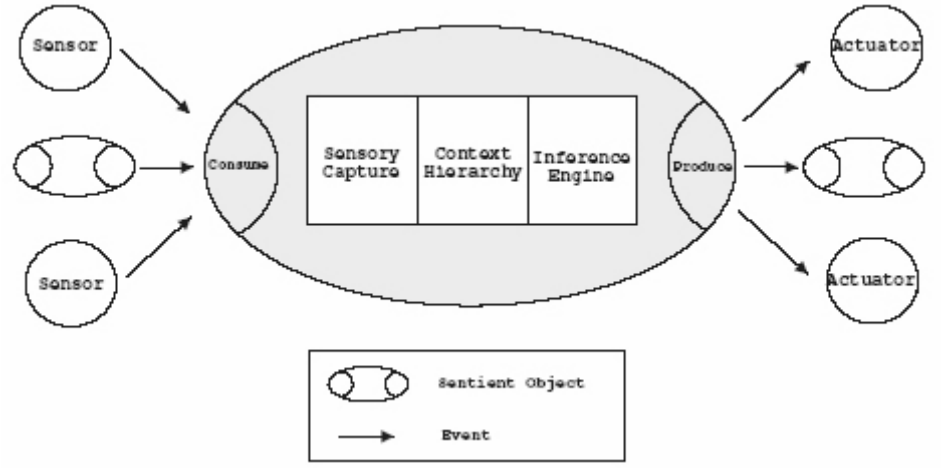

Figure 2.2. The sentient object model, in which actuation is controlled based on sensor input (Biegel and Cahill, 2004).

The sentient object model defines context as any information sensed from the environment which may be used to describe the situation. The sensory capture and context fusion component manages the uncertainty of sense data being consumed. Biegel and Cahill (2004), based upon Bayesian networks, employed a probabilistic sensor fusion mechanism for measuring the effectiveness of derivations of context from noisy sensor data. The context hierarchy summarises the information about the activities to be undertaken and possible future situations in contexts. The advantage of having a hierarchy in the above model (Figure 2.2) and by associating the activities to commence in each context is that “a sentient object’s behaviours are influenced by context” (Biegel and Cahill, 2004, p. 363). The inference engine component is responsible for changing the application’s behaviour, according to the present context. Additionally, the inference

engine has a built-in language which when given a set of facts, and predefined rules, is able to decide which rule to fire, giving the model's intelligence and the ability to react to changing contexts.

The specifications required for handling contextual information in a context are required to detect and determine what actions to take, based on the information discovered. Dey et al. (1999a) presented the Context Toolkit; an infrastructure solution to help in modelling context and its relation to building applications. The context infrastructure model consists of sensors, widgets, interpreters and a server. Figure 2.3 shows the relationship between the context components and the application. The context widgets capture single pieces of context. A context server is very similar to a widget in that it supports the same set of features that a widget supports. The server is responsible for the aggregation of context and for subscribing to every widget of interest, acting as a proxy to the application. The aggregation facilitates the access of context by applications interested in multiple pieces of context about a single entity. Context must often be interpreted before it can be used by an application. The role of the interpreter in the model is to abstract pieces of context not sensed immediately by the widgets. The advantage of having this layer of abstraction is that context will often be interpreted before it can be used by an application. In previous cases the interpretation of context has usually been performed by applications. However, by separating the interpretation abstraction from the application, this will allow for the reuse of interpreters by other applications. Additionally, the model supports interpreters that can be called on by widgets, servers, applications and even other interpreters. When viewing the model as a whole and then breaking down the context infrastructure model into parts, the following conclusions are drawn:

- there is a division of sense data in terms of how to sense and access contextual information;
- the aggregation of context occurs in terms of high-level information rather than low-level details; and
- finally, the abstraction and use of context data is required for the application.

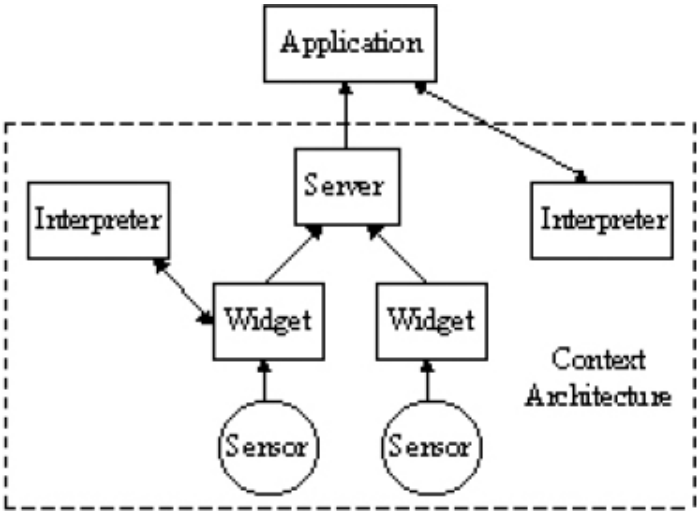

Figure 2.3. The relationship between the application and the context infrastructure model (Dey et al., 1999b)

The Cooltown (Kindberg et al., 2002) model embeds contextual information within a Web-based framework, associating each entity – person, place or object – with a description retrievable via URL. A simple location-based discovery mechanism serves as a form for context retrieval. This involves the use of a wireless beacon to transmit the URL of the local context. From this, other people, devices and objects present in the context can be discovered. The context model is very informal, as arbitrary information can be embedded within the web pages. This feature, together with the restrictive discovery mechanism, is based on the assumption that only information about the local environment is required at any time, limiting the utility of the model. Specifically, the Cooltown (Kindberg et al., 2002) model is suitable as an awareness and information retrieval tool, useful for applications such as museum and tour guides, but not as a more general platform for building context-aware applications.

Judd and Steenkiste (2003) applied a more fragmented approach than the previous solutions. They assign each type of contextual information to a distinct context service. In order to simplify the task faced by applications in interacting with these services, they proposed the use of generic service interface. The query facility offered by the interface is relatively powerful, allowing the client to place a time limit on the response and to specify bounds of meta-attributes, including accuracy, confidence, update time and sample interval (Judd and Steenkiste, 2003). The data model used internally within the services is not specified; this is implementation dependent and related to the types of characteristics of contextual information. There are several disadvantages to this approach. Firstly, users are required to explicitly specify the names of the services in queries. The set of services may be large and dynamically evolving, this places a considerable burden on the user to perform interpretations on the discovery. Secondly, the highly fragmented

view of context that results from the use of many separate services requires users to carry out multiple steps to satisfy all but the simplest of queries and leads to clumsy solutions for representing relationships. Judd and Steenkiste (2003) proposed the use of additional services to capture these.

Finding a way to represent location readings in context discovery that facilitates storage, communication and interpretation is also a challenge. Sensors tend to report locations as a numerical coordinate system (x, y, z), but semantic representation is more effective for application-level reasoning about context (Hazas et al., 2004). Semantic representations usually include a hierarchy of locations such as building, floor, room or country, city, state, and like numerical representations, must also include a notion of uncertainty.

Location is an important aspect of context awareness for mobile users, such as finding the nearest resources, navigation, locating objects and people. Location in a context-aware application requires the modelling of the physical environment and the representation of the location (Dey, 2001; Hazas et al., 2004). Numerous location models have been proposed in different domains and can be categorised into two classes (Mantoro and Johnson, 2003):

- hierarchical – topological, descriptive or symbolic, such as a room; or
- Cartesian – coordinate, metric or geometric, such as GPS.

According to Mantoro and Johnson (2003), the hierarchical location model has a self-descriptive location representation. It decomposes the physical environment into different levels of precision. Additionally, Cartesian location uses a grid on a physical environment and provides a coordinate system to represent locations.

Some context-aware applications require both types of representations. At the spatial dimension, there are some devices such as GPS-based mapping systems, for which the exact Cartesian position in 2D or 3D space is important in defining a sense of absolute physical location. If location information is sufficient in understanding position, the location is considered in relation to other existing objects or sensors (Hazas et al., 2004; Mantoro and Johnson, 2003). Moreover, map-like coordinates are needed to perform Euclidian distance calculations, while information about terrain, traffic patterns and other factors are important to achieve accurate travel time projections. In an 802.11b wireless environment, the location of mobile devices can be determined at the spatial precision of a group of three or four individual offices, by measuring the signal strengths of a few most visible access points (Borriello and Deshpande, 2002).

2.1.3 Context Abstractions and Application Use

Context abstraction and application use is the final phase in the context-aware lifecycle. Such abstractions are essential for the interoperability, usability and development of context-aware systems and location-aware applications. Context-aware computing includes many issues related to the user experience, such as privacy preservation and its associated legal and ethical implications, the questions of usability and user acceptance, and the need for security in the determination and transfer of personal data.

Abstractions of context that simplify the task of describing context-aware behaviour have been proposed independently by Dey et al. (2001) and Ranganathan et al. (2002). They both developed specific context-aware applications that include a reminder tool called CybreMinder (Dey and Abowd, 2000) and a chat program called ConChat (Ranganathan et al., 2002). However, their applicability is much broader. The situation-based model of Dey et al. (2001) is founded on the observation that the context component abstraction of the Context Toolkit (Salber et al., 1999) described in Section 2.1.2 is not the right abstraction for all tasks. To overcome this problem, Dey et al. (2001) proposed to allow programmers and users to define their own high-level contexts in terms of conditions upon one or more attributes. This will allow it to not have to specify the interactions with the widgets, interpreters and aggregators in gathering the required types of contextual information. Situations are formed from a list of equalities: *location = room 201*, and inequalities: *price > $65*, implicitly connected by logical combination. This model is very restrictive, as it is unable to express:

- alternatives – *location = room 201 or location = room 202*;
- relationships – *X is near Y*; or
- uncertainty – *location is possibly room 201*.

The model of Ranganathan et al. (2002) overcomes some of these limitations and discovers contextual information in terms of predicates of the form:

- Context [<ContextType>,<Subject>,<Relator>,<Object>]

As the relator is not restricted to the equality and inequality operators, arbitrary binary relationships can be expressed, such as:

- Context[location, Cathy, entering, room 201]; and
- Context[lighting, room 201, is, off]

There is an implication that historical contextual information and some types of uncertainty have no straightforward representation. An example of historical contextual information is *at 10:35am the location of Cathy is room 201*. Alternatively, an example of uncertainty is *the location of Cathy*

is room 201 with the probability > 0.8. The treatment of unknown or ambiguous information is additionally not addressed.

The investigation of programming tools and abstractions for context-aware systems has not been a priority compared to the development of infrastructural support. The logic-based context abstractions have shown great promise, serving as useful methods for describing contexts in high-level terms. Their utility has already been demonstrated with regard to two distinct application types, namely reminder tools and chat programs, but it is potentially much broader. Amongst the benefits of these formal abstractions is the ability to:

- perform logical reasoning about contexts; and
- combine contexts using the logical operators in a manner not possible with the context models previously discussed, therefore deriving increasingly complex context descriptions

The two applications explored in conjunction with the models – CybreMinder (Dey et al., 2001) and ConChat (Ranganathan et al., 2002) – only scratch the surface in terms of these benefits. In order to fully realise the potential of these abstractions, refinements are required in order to support richer types of contextual information, including relationships, dependencies and historical information, and to accommodate uncertainty. There is also a requirement for improved understanding of suitable development methods that can be applied in conjunction with these models. To date, very little emphasis has been placed on this issue. Some researchers have enumerated the design and implementation tasks involved in developing software, using models such as Dey's (2001) in relation to his Context Toolkit. However, the tasks are almost always described informally and without adequate guidance in terms of the process or complex issues that are involved.

Dey et al. (1999b) acknowledged that the context rapidly changes in situations where the user is mobile and has to adapt to provide "relevant services and information to the user" (Dey et al., 1999b, p. 21). Context-aware computing applications examine and react to a user's changing context in order to help promote and mediate people's interactions with each other and their environment. The knowledge of context aims to be utilised to automate tedious tasks with a certain level of intelligence. Hence, users only need to put a minimum amount of effort into the tedious tasks and concentrate on their other important tasks. In addition, devices that are aware of a user context have higher levels of usability simply because these devices provide interfaces that are able to adapt to the user instead of forcing the user to adapt to the interface. This additional knowledge results in a changed interaction between the user and the mobile computing device. Context-aware services that exploit information about the user and device context are

becoming one of the core components in the ubiquitous computing environment. In order to enable context-aware services, it is a prerequisite to gather contextual information about the device capabilities and the user preferences, and to transform them into user/device properties that can be used in a systematic way for providing context awareness.

2.1.4 Context and Location-aware Applications

Determining the location of people and objects has been the focus of much research in the context awareness and ubiquitous communities. Location-aware applications are numerous in context-aware computing. Examples include portable memory aids, conference assistants, environmental resource discovery and control, support systems for the elderly, tour guides, augmented reality, mobile desktop control, 3D mice, and virtual buttons. Each demands different levels of service from the supporting systems, for example in terms of location accuracy and update rate. There has also been a recent focus on location-aware platforms, which link data-gathering systems and the data-consuming applications in a flexible manner. Such work includes location representation, sensor fusion to combine location data from many sources, and software frameworks supporting the distributed nature of location-aware computing.

The Active Badge system from the Olivetti Research Lab is generally considered to be one of the first context-aware applications (Want et al., 1992). The system is designed so that people can be located in an office and calls forwarded to the closest phone. The office personnel wear badges that emit a unique infra-red signal every 10 seconds, which is detected by sensors placed around the office building. A central location server polls these sensors and the receptionist is able to locate the person and direct the call to the closest appropriate telephone. The system also includes commands to obtain the current location of a badge. The reason for this is to find which other badges are in immediate proximity to a named badge and to find which badges are currently near a specified location. A parallel to draw from this system will be to only disseminate the information that is required to a specific location for the user. Additionally, by reversing the method developed by the research lab, the user can be informed by the system that they are in close proximity to others with similar contextual information and preferences.

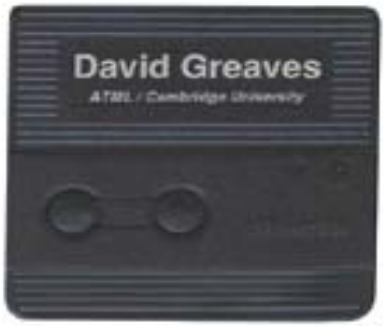

Figure 2.4. The Active Badge emits signals to sensors to locate people in offices (Want et al., 1992).

Context awareness also has compelling uses with the domain of information retrieval. Context dependent information retrieval has been studied both in a general sense and as a basis for the construction of a variety of context-aware guides (Brown and Jones, 2001). Two of the many known applications are GUIDE (Cheverst et al., 2000) and CyberGuide (Abowd et al., 1997). Despite its high similarity with the Active Badges system (Want et al., 1992), both applications present tourists with location-dependent information and services on handheld devices. The GUIDE application provides visitors to the city of Lancaster with information about nearby attractions, tailored city tours, interactive services such as hotel booking facilities, and a messaging service, all within a Web-based navigational interface.

The CyberGuide application explores the provision of similar services to single-building and campus-wide settings (Abowd et al., 1997). Started as an indoor mobile tour guide service for visitors to the Graphics, Visualisation and Usability (GVU) Centre at the Georgia Institute of Technology, it extends its ability to provide outdoor mobile tour guide services as well. The CyberGuide project focuses on how portable computers can assist in exploring physical spaces and cyberspaces. It is an intelligent tour guide that provides information about the surrounding environment, such as the physical layout of the local area, the current position of the user, and nearby points of interest. A variety of context-aware guides have since been constructed for museum, exhibition and conference environments. However, after a revision of the original implementation, the use of static configuration causes a detrimental effect on its evolution and extensibility.

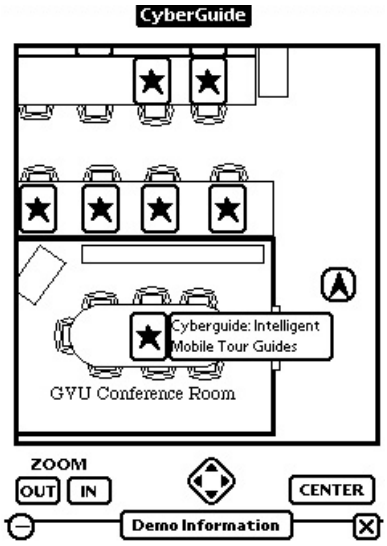

Figure 2.5. The CyberGuide is an intelligent tour guide which assists in exploring physical spaces (Abowd et al., 1997).

The ParcTab is a system developed at the beginning of the 1990s at the Xerox Palo Alto Research Centre (Chen and Kotz, 2000). The ParcTab acts as a mobile personal digital office assistant. The system is based on palm-sized wireless ParcTab computers and an infrared communication system (Figure 2.6). It links them to one another and to desktop computers through a local area network. The system developers carried out experiments with ubiquitous computing and context awareness in an office environment. The ParcTab system can continuously know where each ParcTab is – which ones are in the same room – and notify applications of location changes. Improvements can be made on ParcTab's use of context by identifying the presence of other mobile devices to communicate with its users and other non-mobile objects in the environment.

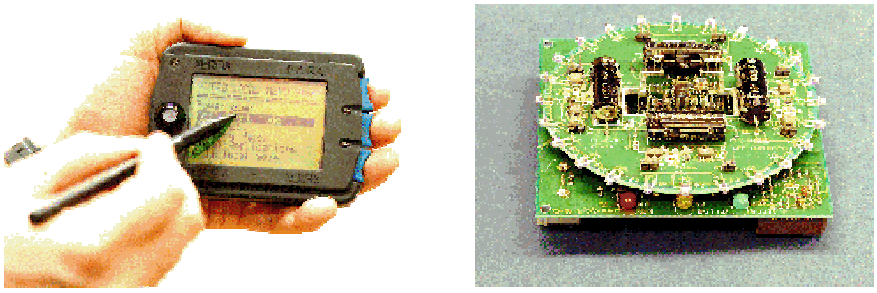

Figure 2.6. The ParcTab computer (left) communicates with infrared sensors (right) (Want et al., 1995).

Salber et al.(1999) also identified these shortfalls as the basis for their work on the Context Toolkit. Both Active Badges (Want et al., 1992) and CyberGuide (Abowd et al., 1997) are quite

similar in the way that they both focused on a specific type of context, namely location. Another similarity, which is not so obvious, is that the system's architecture is decomposed into smaller components. While the Active Badge system consists of four layers: network control, representation, data processing and display interface, the CyberGuide decomposes itself into four different components based on the role each of the components is required to fulfil. In a research paper by Dey and Abowd (1997), they stated that the greater the awareness of the user's context, the better it is able to support the user in their task. Although this statement is vague, as there is no operational definition of *context*, it does indicate how important context discovery infrastructure is in the context-aware lifecycle.

Perhaps the most ambitious use of contextual information is demonstrated by a variety of projects that are concerned with the embedding of computing and context sensing infrastructure into environments such as homes, classrooms and meeting rooms. The goals of these projects are diverse. Several aim to monitor activities of the elderly within their homes or assisted living settings, with the objective of allowing them to retain their independence while ensuring emergencies are detected (Beckwith, 2003; Intille et al., 2002; Mynatt et al., 2000). Others use contextual information to automate the control of environmental conditions, including ambient lighting, temperature and various other appliances present (Hirsh et al., 1999). Finally, the classrooms and meeting room projects are principally concerned with facilitating remote collaboration or with providing integrated capture and note taking facilities, enabling rich records to be retained of the proceedings that occur within the specially instrumented contexts (Hannon et al., 2005).

The Conference Assistant is a prototype application developed by Dey et al. (1999b). It is used to aid people attending conferences and will help them with concurrent activities that occur whilst in attendance at a conference. The Conference Assistant provides awareness of activities to the users of the system. It promotes interaction between users and their surrounding environment. The application operates on RFID (radio frequency identification) technology to sense identity and location. The Conference Assistant uses location information, time, activities in a certain location, and user preferences as contextual information (Chen and Kotz, 2000). The application demonstrates a wide variety of context services, such as contextual sensing and adaptation, to provide information to its users in real time. These applications can help users navigate unfamiliar territory and have access to information in the most useful and unobtrusive way. The use of contextual information in applications decreases the human-centred attention an application requests to service the user's needs. Context-aware computing offers an opportunity to increase human productivity.

The Owl application developed and experimented with at the IBM T.J. Watson Research Centre (Ebling et al., 2001), accumulates and retains contextual information from numerous sources. Users can query for current contextual information or submit a query to be notified of when their query conditions are satisfied and the contextual information is made available. There is a belief that knowledge of past contextual information may allow an understanding of present or future behaviour in the absence of current information, therefore reducing the time needed to produce this information to the user-based upon their request. Additionally, it may allow us to pre-fetch contextual information in anticipation of future contexts, providing better performance and availability.

In order to provide planetary-scale user positioning at very low cost, the initial Place Lab (Pescovitz, 2004; Schilit et al., 2003) infrastructure leverages the widespread proliferation of wireless networking access points based on the IEEE 802.11 standard, commonly known as WiFi. Place Lab leverages the urban ubiquity of WiFi access points to bootstrap location-enhanced computing (Pescovitz, 2004). While Place Lab architecture can be adapted to work with most types of wireless technology, including Bluetooth, Wi-MAX and mobile phones, the huge scale of WiFi deployments provides a unique opportunity to add location to many wireless data networks at very low cost (Schilit et al., 2003). Combining novel machine learning techniques with a distributed global WiFi positioning database of wireless access points, also known as hotspots, Place Lab software gives client devices the ability to passively listen for nearby access points and compute their own position. This latitude and longitude can inform myriad mobile applications and services.

The Place Lab global WiFi positioning system uses radio frequency between beacons to determine the location of the mobile device. Key to this approach is the fact that nearly all wireless access points broadcast their MAC addresses, a globally unique 48-bit value that identifies each node in a network. Essentially the access point is constantly *shouting* its name, and this is a name that no other access point in the world shares (Pescovitz, 2004). These beacon broadcasts enable new network clients to see the various networks that may be available in a particular physical location. Even if you are not authorised to connect to that particular network, the WiFi-enabled mobile device will still hear the MAC address and can use that signal in calculating the location (Schilit et al., 2003).

Once a laptop hears the MAC address, the Place Lab software, running locally on your own computer, looks up that unique access point in a locally cached directory of hotspots. The directory maps MAC identifiers to latitudes and longitudes, not unlike the way a phone book links a name with a physical location (Schilit et al., 2003). The signal strength of the MAC beacon is

also taken into account, providing information about how close you are to the access point. In places where the laptop detects multiple hotspots, the intersection of coverage can be used to better pinpoint the user's location. The range of access points is approximately 100 metres, enabling the Place Lab algorithms to estimate location with at least 100 metre precision (Pescovitz, 2004).

The Ubiquitous Walkabout, part of the Ubiquity project in Intel Research, is an investigation into distributed location-aware computing (Light et al., 2003). It involves *information beacons*, which wirelessly broadcast specific information about a location to a small vicinity (approximately 10 metres), and a *personal server*, which can receive the beacon messages and process them as they are received. Walking down a street equipped with this technology, a user would receive the information from many information beacons in turn, each of which will transmit information about a specific service or other offering in its vicinity. Based on previously expressed policies and preferences provided by the user, the personal server might report the service or offering to the user, respond to the offering automatically, log the information for later use, or ignore it (Light et al., 2003).

The personal server is a capability that can be part of any small mobile device, such as a mobile phone or PDA, and can run continuously for a long time (one or more days), providing considerable computational and storage resources to its user during that time (Want et al., 2002). It uses advanced power management capabilities to provide the appearance of being *always on*, though it may in fact enter sleep modes from which it can quickly return. In its current form, the personal server communicates with the world through one or more wireless (radio) interfaces (Light et al., 2003; Want et al., 2002). It can talk to PCs, public displays, or mobile phones.

An information beacon can be as simple as a radio chip, microcontroller, and power supply. It need only be able to store a small amount of information that it broadcasts repeatedly, and can be manufactured for under $20 in large quantities. To establish the content of a beacon, it can simply be plugged into a standard computer. Currently, Light et al. (2003) use Berkeley Motes as information beacons. They use a simple radio technology to broadcast a short (300 byte) message repeatedly. More recently Light et al. (2003) have used mote technology based on a Bluetooth radio (iMote) to provide larger messages.

Ubisense was founded in January 2003 by the team who developed the Bat ultrasonic location sensor and the SPIRIT distributed middleware platform (Addlesee et al., 2001) at AT&T Laboratories, Cambridge. Ubisense has developed an in-building ultra-wideband (UWB) radio-based tracking system which can determine the positions of people and objects to an accuracy of

a few tens of centimetres, using small tags which are attached to objects and carried by personnel, and a network of receivers which are placed around buildings (Cadman, 2003). Ubisense has also developed a scalable middleware platform which can manage and distribute large volumes of real-time location information to very many clients, and which simplifies creation of location-aware applications

Ubisense is targeting its sensing and middleware technologies to a number of markets, including healthcare, security, workplace productivity and military training. The end applications for location-aware technology in these markets are varied, and range from ensuring that a baby is not removed from a care facility by anyone other than its mother, to making sure that a *simunition* (an explosive charge used in military training) does not detonate if a soldier happens to be standing too close (Cadman, 2003). However, I believe that these apparently disparate applications have much in common, and that proponents of context-aware computing must focus on these common elements if the field of location awareness is to move beyond its research roots.

Cadman (2003) provides examples of potential location-aware applications with which end-users have approached Ubisense. These include:

- maximising productivity of a hospital environment;
- visitor management in security-conscious sites;
- fire fighter training; and
- military training in urban combat scenarios.

Each of these applications represents a problem that is both pressing and unsolved, and where location awareness is a valuable system attribute. Clearly, technology choices will have to be made based on the demands of each application and the scenario in which it is to be deployed, but the technology plays a secondary, supporting role to the application.

In particular, high location accuracy will be important for many in-building applications – as the accuracy of their tracking system increases – location-aware applications become more robust, increasing end-users' confidence in them. Accuracy specifications for location sensors are often misleading and hard to interpret, making comparisons between systems difficult. However, examining the accuracy of a sensor at a 95% confidence level provides a good indication of how a system will work in practice (Addlesee et al., 2001; Cadman, 2003). At this level, the vast majority of sensor readings will fall within the quoted specification, allowing system designers to match an application to a sensor technology with confidence that end-users will perceive the system as being robust and trustworthy (Addlesee et al., 2001).

UWB seems well-suited to in-building location-aware applications, because of its non-line-of-sight nature, modest infrastructure requirements and high tracking accuracy. A properly architected UWB tracking system is low-power (thus low-maintenance) and the fundamental technology is simple and low-cost. This latter factor will also improve as UWB technology becomes more widely deployed, such as in communications products for example (Cadman, 2003).

The MIT Cricket indoor location system was motivated by the importance of mobile and context-aware applications in pervasive computing environments, and the poor indoor performance of GPS (Balakrishnan and Priyantha, 2003). Some of the goals were to build a location-sensing technology that would:

- scale well to large numbers and high densities of devices that needed location information;
- make it difficult to track users, thereby helping with the user privacy problem that plagued previous location systems (such as Want et al. (1992) Active Badge system (Want et al., 1992)); and
- be easy to deploy and manage in large buildings.

These goals led to an architecture that is radically different from other indoor location systems like the Active Badge (Want et al., 1995) or Active Bat (Addlesee et al., 2001). In the Cricket project, ceiling or wall-mounted active beacons send periodic chirps of RF and ultrasonic signals. Passive listeners, connected to host devices (PDA, laptop, sensor, etc.), estimate distances to individual beacons using the standard time-difference of arrival technique. This architecture *inverts* the architecture of the Active Badge and Bat systems, which use passive ceiling-mounted receivers that obtain information from active transmitters attached to devices carried by users.

Qualitatively, the Cricket architecture offers the following advantages over the Active Badge and Bat architecture (Balakrishnan and Priyantha, 2003):

- good scalability – RF and ultrasonic channel use is independent of the number of devices in any region, when host devices actively transmit, high-density deployments are harder to achieve;
- ease of deployment – Cricket beacons are easy to deploy and do not require any infrastructure connecting back to a base station, they can be placed with few constraints inside rooms or corridors; and
- user privacy – Cricket's architecture allows a host device to infer its location without the infrastructure or any other entity learning that information. While Cricket by itself cannot guarantee user privacy, it makes centralised tracking of users difficult.

However, these advantages come at some cost and this research identifies the following problems:

- continuous tracking is difficult – a listener hears only one beacon at a time, updating the position of a moving device is more complex than in a system that simultaneously obtains multiple distance estimates from the device to known positions;
- beacon scheduling – Cricket requires a distributed beacon scheduling scheme that avoids collisions at the listeners; and
- energy consumption is potentially higher.

Many lessons have been learnt from the Cricket project. Initially, the project was interested in context and resource discovery, for which information about the space in which a user or device is in is important. Providing spatial information requires good boundary detection. However, over time, it was realised that the users wanted to build navigation applications, which require knowledge of position coordinates and orientation in some coordinate system, in addition to spatial information. This led to the project implementing algorithms to provide these two additional forms of location information. Cricket version one provides information about space, position and orientation (using the Cricket compass).

Still in development, Cricket version two fixes several shortcomings of version one based on experience with several applications. First, because Cricket version one was primarily optimised for good spatial boundary detection, its position accuracy in real deployments had high variance, being accurate to only about 30–40 cm. Cricket version two improves this significantly, being able to obtain distance estimates to within 1 cm on average and 3 cm most of the time. This is achieved by using better ultrasonic signal processing, a better outlier rejection method to filter out bad distance samples, and by implementing finer-grained sample timing at the listener (Balakrishnan and Priyantha, 2003).

The Cricket version 2 software also incorporates a “single constraint at a time Kalman-filter tracking algorithm” (Balakrishnan and Priyantha, 2003, p. 8) that allows moving devices to be tracked without requiring an active transmitter on the device. Preliminary tests of this algorithm indicate that its performance at pedestrian speeds compares well with a system that simultaneously obtains multiple distance estimates of the device to known positions and performs a least-square minimisation on the resulting linear simultaneous equations.

The biggest lesson from user applications is that there appears to be no single killer location-aware application for indoor environments. Several potential applications – across a variety of domains – fall into the following categories:

- indoor navigation for robots and people;
- games and virtual realty;
- asset tracking;
- content redirection (music, video, desktop); and
- embedded sensor network applications.

To support this wide range, the underlying platform must provide as much useful information as possible to applications.

The mobile phone manufacturer Nokia has introduced the Nokia Sensor[1], a mobile phone software application that offers a new, exciting way for people to create information and share it with other mobile users nearby. Using the Nokia Sensor application, users can create personal pages on their phone called *folios*. A folio is a short-range mobile homepage that a user creates. It houses your avatar, interesting conversation topics and contains details about all quirky habits and unusual hobbies. The Nokia Sensor application works over Bluetooth wireless technology, providing connectivity within a *circle* of up to 10 metres from the mobile device. When other Sensor phones come within this circle, users can also check out the folios of other Sensor users in their vicinity, exchange messages and share information (photographs, personal video and audio files), write messages in the Guestbook or just read the profile filled in by the owner. There is also the possibility of setting alerts according to who shows up in the network, Group Codes which set the personal page to alert when someone with similar hobbies and interests shows up in the area. The guestbook allows visitors of the folio to write messages or comments, and according to the counting function, it will point out precisely the number of users who have viewed your page, this being the criterion for establishing the popularity of a user. The Nokia Sensor application creates a totally new way of communicating with people in the same location, for example, in cafés, get-togethers, buses and trains. People are able to use their creativity and imagination to learn more about other people and their interests. It boosts the opportunities of Bluetooth wireless technology on the mobile device. Nokia Sensor is a spontaneous, sociable application for spontaneous, sociable people. The Nokia Sensor application allows users to express themselves and discover new things about others nearby in a novel and simple way. This means that the value of using mobile devices is expanded to connecting with those who share the same space at the same time, acquainted or unacquainted. This application will show its true potential only in a network with lots of users.

[1] The information described this paragraph is sourced from http://europe.nokia.com/A4144923. Additionally, there is a Flash presentation demonstrating how the Nokia Sensor application works in a social context.

2.1.5 Overall Application Analysis

All of these application examples illustrate how much context awareness has contributed to advancing users' adaptation to ubiquitous computing. However, one of the main challenges in pervasive and context-aware computing is the ability to handle uncertainties that emerge when applications try to become aware at runtime of desirable situations and are indecisive in reasoning about the true situation. Three factors have been observed by Padovitz et al. (2004a) that promote context uncertainty and highlight the need for context verification. The first is an unsatisfactory combination of attribute types (either virtual or physical) to infer a desired context. The second is the intrinsic ambiguity between two or more contexts that impedes a straightforward reasoning about the correct context. This is often the case when two different contexts are characterised by two similar attribute values. The third factor is often the inherent inaccuracy and unreliability of many types of low-level sensors, which may lead to contradictory or substantially different reasoning about a context. When faced with contradictory sensorial data from two similar sensors, a context-aware system needs to resolve these discrepancies as well as high-level context ambiguities that result from the contradictory sensor readings.

The central scheme in a verification of context is the ability to resort to other contextual information that will help adjust sensor attribute values (Padovitz et al., 2004b). The verification process assumes the correctness of a specific possibility and determines its probability according to other contextual information. It then switches and assumes the correctness of other possibilities and assigns a probability to each as well. Finally, it selects the most probable alternative. An example by Padovitz et al. (2004b) described two identical light sensors in a room yielding opposite readings; the first indicates the light is on and the other indicates the light is off. By resorting to other elements, such as the time of day, motion in the room, computing or other activities occurring in the room, the system can assign probabilities to both readings and select the most probable. Another example might concern when a person is detected as being located in two places at the same time by observing two similar location detector devices, which is their PDA location versus their electronic badge location. In this case, the system returns to other contextual parameters which will indicate the most probable location value.

The need to verify low-level context (sensor readings) is not only required when two or more similar sensors yield different or contradicting results, but is useful when pervasive systems deal with sensors that are inherently inaccurate. Minimising inferred location errors takes high importance when relatively short distances imply a totally different context for context-aware systems, for example, the varying context interpretations for different places in a building. The context *Meeting Room* is completely different from the context *Office*, even though some

locations in the *Meeting Room* and *Office* are in close proximity, and are only separated by a structural element like a wall.

Context-aware applications typically assume that the contextual information upon which they rely is complete and accurate. However, this assumption is usually unjustified, as sensed contextual information is often inaccurate or unavailable as a result of environmental noise or sensor failures, while user-supplied information is subject to problems such as human error. In order to support effective reasoning about contextual information, it is important to distinguish between false and unknown information. An infrastructure that manages contextual information from a variety of sources and facilitates the exploitation of this information by applications is advantageous. The reasons and responsibilities are twofold. First, the context gathering components discover and process sensed contextual information. Second, a distributed context management system assumes the role for integrating, storing and managing contextual information from a range of sources, responding to context queries, and generating notifications of significant context changes. The context management system also provides interfaces for users to easily update and browse their static and profiled contextual information. In the case of sensed contextual information, these are responsible for mapping the heterogeneous contextual information produced by the context discovery components into the fact abstraction used by the context management system, routing queries from the management system to the appropriate components, and detecting and resolving conflicts and ambiguity where possible.

Context-aware applications exploit information about the context of use, such as the location, tasks and preferences of the user, in order to adapt their behaviour in response to changing operating environments and user requirements. This information is gathered from sensors or human users. It is not known ahead of time when applications will require certain pieces of contextual information. There is a need for them to be persistent and available at all times. This is linked to the need for execution persistence and the request to maintain historical information and memories of context. In general though, if a more complete memory system is required, it is left up to the application to implement it. There may be an application in the future that requires the history of uncertain contexts and lends itself to a memory-based implementation to aid in this requirement.

Section 2.2 discusses the models of memory and how context-aware applications may benefit from learning about context and recalling it as and when required. It is foreseen that a constructive memory and reasoning model can be integrated as a component in context awareness.

2.2 Memory

Learning and remembering are concerned with more than storing and retrieving something. A psychological view of memory might be styled as knowledge of an event or fact which may have not been thought of, with the additional consciousness that it has been thought or experienced before (James, 1892). James (1892) viewed an idea as a process; the experience of remembering is the experience of being aware that a reactivation or recomposition of a previous experience is occurring. That is, the mere construction of a recurrence is different from the human experience of perceiving that this is the same as what has been experienced (or done) before. This awareness of the past is better called remembering. Remembering is not the re-excitation of innumerable fixed, lifeless traces, but rather it is an imaginative reconstruction or construction, built out of a relation of the attitude towards a whole active mass of organised past reactions or experiences (Clancey, 1997). Remembering is a conscious process of recollecting a previous experience. This process is imaginative and involves an emotional attitude about the experience. What is constructed is the relation between past experience and detail in the present experiences. The past experience exists as a whole active mass and is organised. Reconstructing experience involves establishing a new physical coordination which reuses previous perceptions and conceptions. Hence, remembering is not the retrieval of one thing but the re-establishment of a relation and way of coordinating perception, words, ideas and actions.

There is the obvious observation that humans can recall things such as people's names, contact telephone numbers, lines from a movie or lyrics from a song. Although empirical evidence clearly shows recall is bettering contexts similar to the context where the information is acquired, recall of the de-contextualised information is possible and even desirable in problem solving. According to Clancey (1995), one can only know what is perceived. Constructing a representation (context) means seeing something in a new light. In this statement, representation is defined as a combination of percepts and memory. As an example, each act of speaking is a complete act of perceiving in itself. By speaking, new meanings are created which are perceivable by ourselves and others, and consequently open to reinterpretation. Each memory contains what is retained from a previous activity and is capable of recalling previously enacted sequences of behaviour.

2.2.1 Constructive Memory

Recall of memory is based on matching between specific features and indices contained within the context or environment. An index (or set of indices) used to retrieve a context event is not modified as it is used during the retrieval process. A constructed memory may or may not contain the same information used initially to cue the system. What gets constructed is not a function to cue the system but depends on what the system has learnt to be associated with proposing

alternative experiences as a basis for interpreting the environment (Liew and Maher, 2004). Situatedness contains the two fundamental ideas of interaction and construction (Gero, 1999). Interaction implies that what the systems knows is not encoded priori and indexed for use later, but rather that knowledge is developed through interactions with the environment. This interaction interprets the experiences processed by the system according to the current situation in such a way that the interpretations of past experiences are filtered defined by the present situation. Compared to non-situated systems they lack this interaction. The construction of memory is about previous experiences providing the basis for interpreting the past, according to the current situation (Gero, 1999). A past experience is not copied into the present but rather interpreted based on the current interactions with the environment. Once interpreted, it eventually becomes part of the memory system – as a new experience – that influences subsequent interpretations.

A memory is not a piece of information that resides in some stored form, indexed and reproduced in the same way each time it is called upon. Rather it is constructed anew each time the need for it arises. Memories are initially constructed from an experience in response to demands for memory of that experience, but the construction of memory includes the situations pertaining at the time of demand for memory. For this reason, the construction of memory in response to a condition in the external or internal environment is a situated act (Liew and Gero, 2003).

Constructive memory (Clancey, 1997; Gero, 1999; Gero, 2003; Liew and Gero, 2003; Liew and Maher, 2004) as a theoretical concept is more concretely known than situatedness. It refers to the property or characteristic of memory in which the memory of the past experiences affects, guides and cues the construction of the current experience, and the current experience in turn changes the memory. In computational terms the current models of memory are indexed-based where exactly the same thing of the past is pulled up as memory; that is, memory remains static and changes only as additions or subtractions occur. However, in human memory, memory is constructive, that is, past experiences affect the current ones and the current ones change the memory of past experiences in a circular way. This leads to more learning and richer behaviour. This is the basic core concept.

Memory is a creative construction or reconstruction built out of the relation of our attitude towards a whole active mass of organised past reactions or experiences, and to a little outstanding detail which commonly appears in an image or language form. It is thus hardly ever really exact, even in the most basic form of memorisation by repetition, and it is not all important that it should be so. Hence remembering is a constructive act. Memory is a label for a diverse set of cognitive capacities by which humans retain information and reconstruct past experiences, usually for present purposes. Humans remember experiences and events which are not happening now, so

memory seems to differ from perception. Additionally, they remember events that really happened, so memory is unlike pure imagination. Memory seems to be a source of knowledge, or perhaps it just is retained knowledge. Remembering is often suffused with emotion and it is an essential part of much reasoning. Some memories are shaped by language and others by imagery.

A constructed memory defines both the interpretation of the relevant content and the interpretation of the environment, despite the interactions taking place between the system and the environment. The model of constructive memory (Figure 2.7) exists within the structure of situated computing (Gero, 2003). The important characteristic of constructive memory is the ability to incorporate the previous experiences with the construction of new and different memories. New memories result from the recursive interpretations of the context and environment with which the system is framed – leading to added memory and experience. After each new memory is constructed, it is available for subsequent memory constructions.

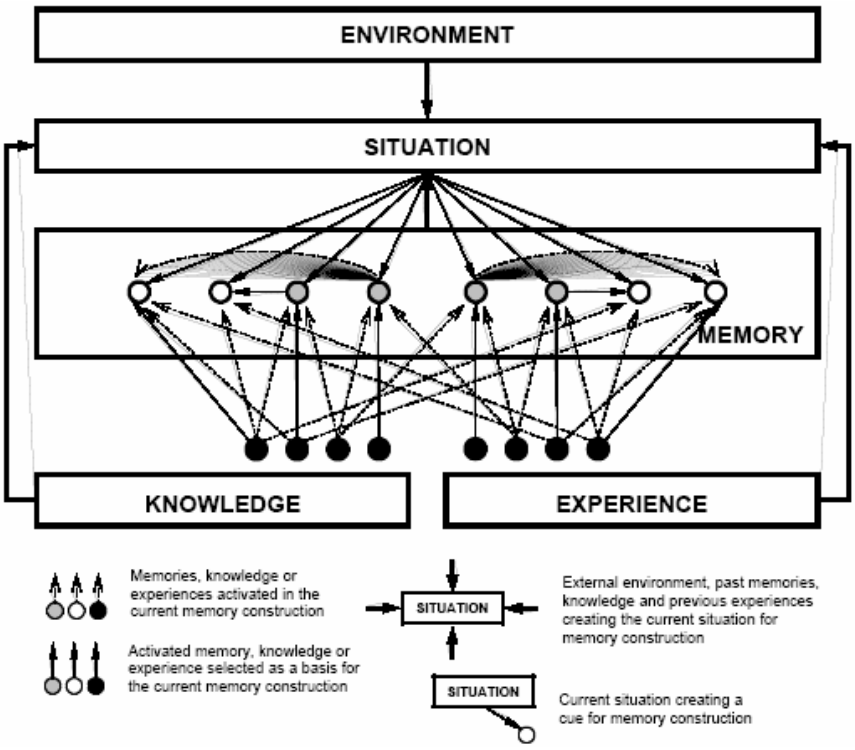

Figure 2.7. Constructive memory model (Gero, 1999).

Memory construction occurs whenever the system uses a past experience in the current environment in a situated manner (Gero, 1999). Any information available in the current environment is used as cues in the construction process. Associated memories are constructed and grounded according to the current interactions between the system and the environment.

This behaviour of a constructive memory system within a situated environment has the following characteristics (Liew and Gero, 2003):

- memory operates as a dynamic process, it is not a static imprint to be stored in a specific location and retrieved for use later;
- the operations of the memory system are not predefined, they are influenced by the situatedness;
- construction of memories does not rely on the exact matching between what the system has in the memory and the current designing environment; and
- memories that are constructed may not match the original experience exactly as it was first experienced, but change according to when, where and with what the memory system is cued.

This process is a contradistinction to memory as a retrieval process whereby there is a memory stored that can be retrieved directly and the retrieval has no effect on the memory.

Consider MIT's remembrance agent (Rhodes and Starner, 1996) which operates in a note-taking application from where it monitors the themes the user is currently entering and offers links back to previous notes with similar themes. It is possible to imagine a future with a situated remembrance agent embedded in a mobile computing device that is able to cross-reference different media files captured in related situations or with related contexts. The situatedness of the application provides an opportunity to offer unexpected value while the user is primarily engaged in an activity. As we enhance situations in the physical world with relevant connections, our experiences will be greatly enriched.

Current uses of the term memory involve tacit or explicit assumptions. Memory can be broken down into a set of memories that consist of relatively independent traces stored in some location. These traces must be searched for and recalled in order to produce remembering, and appropriate traces must be excited in order for past experiences to have their different effects on subsequent events. The key point is that experience cannot be viewed as isolated stimuli; rather, what is experienced is a construction of the quality of the event, the result of the interaction of the experiences and the world (Jenkins, 1974). What is constructed is an organised configuration or pattern of experiences. Recall and recognition experiments designed to test comprehension and memory also result in constructed experiences. Experience is relational within a context conceived by the subject.

In most of the research in context-aware systems discussed throughout Section 2.1, the problem of filtering the vast amount of contextual information that is available, in such a way that the important identification of important constellations of contextual information is feasible, has not

been addressed (Maximini et al., 2003). A promising method concerned with adapting to new situations by remembering similar earlier experienced situations or cases (Zimmermann, 2003) is case-based reasoning.

2.2.2 Case-based Reasoning

Reasoning is often modelled as a process that draws conclusions by chaining together generalised rules, starting from scratch. Case-based reasoning takes a very different view. In case-based reasoning, the primary knowledge source is not generalised rules but a memory of stored cases recording specific prior experiences. In case-based reasoning, new solutions are generated not by chaining, but by retrieving the most relevant cases from memory and adapting them to fit new experiences. The reasoning is based on remembering. Case-based reasoning can also be beneficial when a reasoner must solve problems that are quite different from prior experiences (Aamodt and Plaza, 1994).

Case-based reasoning combines memory with explanation-based learning to allow a program to learn by remembering and analysing, and will search memory for a previous case in which a similar task arose (Leake, 1996). Regardless of whether a case-based reasoner solves a routine or novel problem, and of whether the problem-solving outcome is success or failure, the case-based reasoner learns from its experience. Complementary to the principle of *reasoning by remembering* is the principle that reasoning and learning are intimately connected (Leake, 1996). The knowledge of a case-based reasoner is constantly changing as new experiences give rise to new cases which are stored in memory for future use. A case-based reasoner learns from experience to exploit prior successes and avoid prior failures.

Watson (1997) explained that case-based reasoning is driven by two primary motivations. The first is from cognitive science and its desire to model human behaviour. The second is from artificial intelligence and the pragmatic desire to develop technology to make artificial intelligence systems more effective. Interest in case-based reasoning as a cognitive model is supported by studies of human reasoning which demonstrate reasoning from cases in a wide range of task contexts. Humans are robust problem-solvers; they routinely solve difficult problems despite limited and uncertain knowledge, and their performance improves with experience (Watson, 1997). All of these qualities are desirable for real-world context-aware systems. Consequently, it is natural to ask how case-based reasoning can advance context-aware applications. Defining an initial knowledge base is generally only the first step towards a successful context-aware application. Initial understanding of the problem is often imperfect, requiring system knowledge to be refined. Likewise, changes in task requirements and circumstances may render existing

knowledge obsolete. Although refinement of case representations and indexing schemes may be required as a task becomes better understood, case-based reasoning offers a significant benefit for knowledge maintenance; a user may be able to add missing cases to the case library without expert intervention.

Due to case-based reasoning systems being able to undertake incremental learning, they can be deployed with only a limited set of training, to be augmented with new cases if (and only if) the initial case library turns out to be insufficient in practice (Zimmermann, 2003). A case-based reasoning system for context awareness needs only to handle the types of problems that actually occur in practice, while traditional context-aware systems must account for all problems that are possible in principle. The results of case-based reasoning systems are based on actual prior experiences that can be presented to the user to provide compelling support for the system's conclusions. Successful use of case-based reasoning depends on addressing issues of how to acquire, represent, index, and adapt existing experiences from memory.

AmbieSense (Mikalsen and Kofod-Peterson, 2005) applies cased-based reasoning as a lightweight reasoning mechanism that is capable of running on a small mobile device. The goal is to develop a set of software and hardware tools to facilitate in context-aware computing. Different types of contextual information can arrive in a very diffuse fashion. Since case-based reasoning works on discrete cases, the continuous values flowing into the system must be made discrete. The context agent receives a current context from the context middleware, translates it to fit the ontology used and sends it to the reasoning agent. Once a new context arrives, the case-based reasoning cycle is activated (Figure 2.8). The system will try to retrieve a known context or case, and classify the current situation based on the retrieved one. When the situation has been classified, the associated goal is presented to the task decomposition agent. After the agency has handled the problem, the case will be stored in the case-base, consisting of (Mikalsen and Kofod-Peterson, 2005):

- contextual information describing the situation;
- the problem(s) associated with the situation; and
- the solution(s) constructed by the application agents.

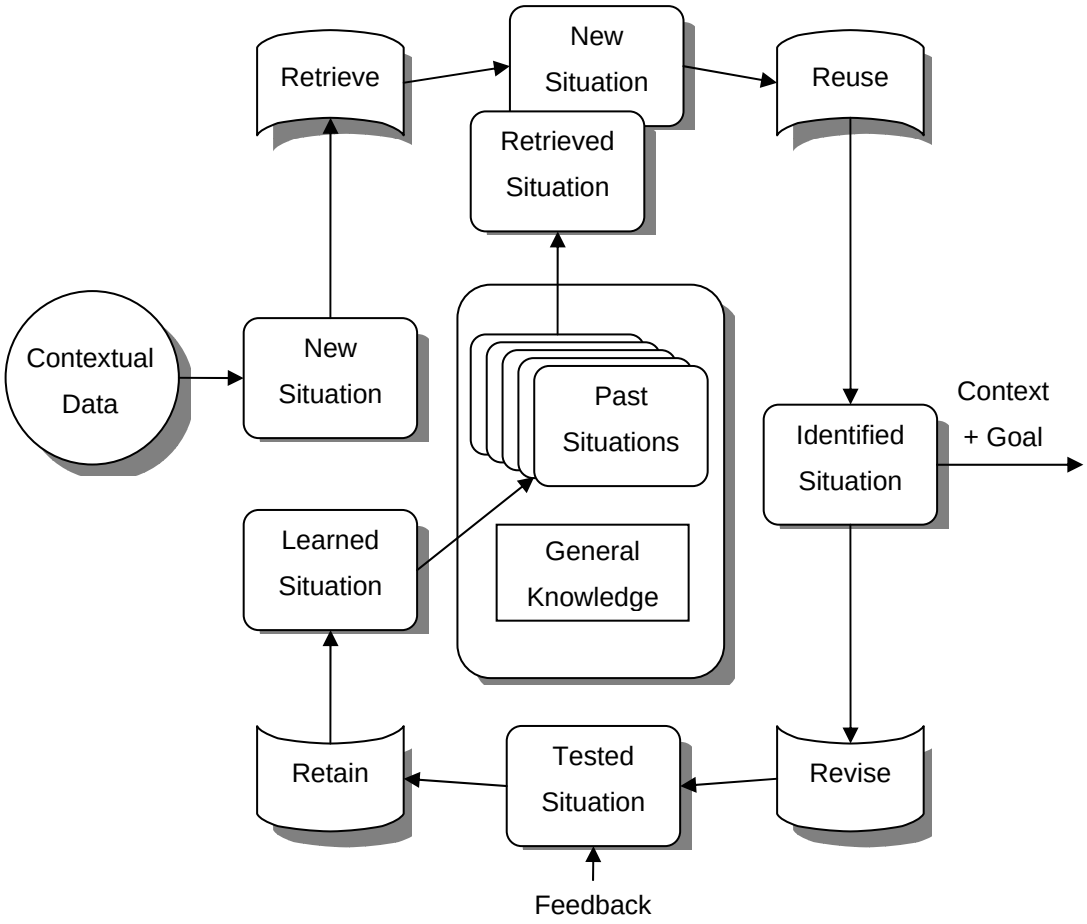

Figure 2.8. Modified context reasoning cycle from Mikalsen and Kofod-Peterson (2005).

There are potentially two main problems with the use of case-based reasoning for identifying situations: the storage problem and the problem of indexing and searching. Firstly, to solve the problem of storing the potentially vast amount of cases constructed during runtime, the user will have persistent storage available and will use it for storing the cases. The large amount of data does not only affect the amount of storage space needed, it will also severely affect the indexing and matching algorithm used. To remedy this, an approach might be to group all similar cases into prototypical cases. As an example, everything that the one hundred meeting cases have in common will constitute the prototypical meeting situation. These prototypical cases will be a part of the case repository and will be used every time in the reasoning process. It is not sufficient to gather and aggregate contextual data without a required amount of reasoning about the data. Research conducted by Zimmermann (2003) suggests that case-based reasoning is a promising method for identifying the correct combination of contextual information which leads to a good situational understanding.

Liew and Maher (2004) used knowledge and experience to provide the basis for memory construction according to the environment and the current situation. One of the advantages of situated case-based reasoning is "the way its knowledge and experience are understood and used" (Liew and Maher, 2004, p. 200). Knowledge and experience are built upon the understanding of the environment and rely on the selected pieces of information based on the situation. Knowledge can be defined as the general pieces of information that are known and recognised through experience. Experience therefore refers to previous events or conditions encountered. These experiences can be in the form of information collected over time under controlled conditions and require the system's direct involvement with that particular situation.

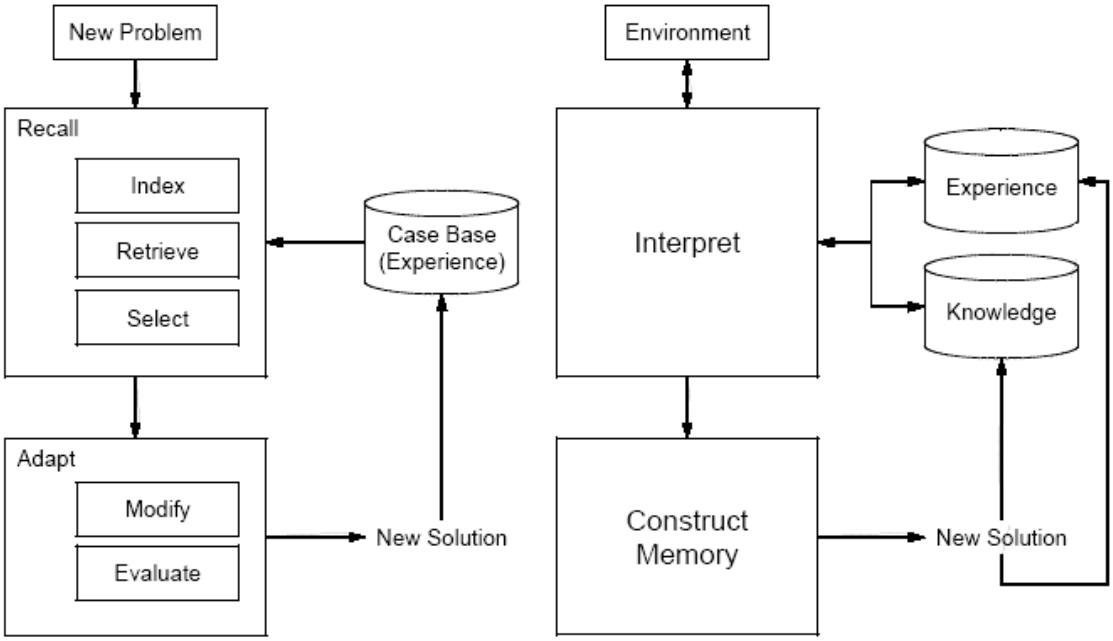

Figure 2.9. A generic case-based reasoning model (left) and a situated case-based reasoning model (right) (Liew and Maher, 2004).

Applied case-based reasoning systems often entirely forgo adaptation. They function solely as memories, retrieving cases and presenting them to the user, who adapts them on their own. The goal of a case-based reasoning system is to generate a useful solution. Normally, this is accomplished by adapting a prior solution for application to a new problem. An alternative method is to adapt the problem situation itself, so that the retrieved case can apply to the new problem without adaptation. At the heart of case-based reasoning is the importance of experiences and lessons of remembering and reusing specific experiences to aid in the sustained learning process.

2.3 Summary

The literature review and background research presented in this chapter have provided insights that have shaped the research presented in the remainder of this thesis. Keeping a history of the user's interaction with the environment is of use for many reasons. However, collecting, structuring, accessing and reviewing such potentially large amounts of information is not trivial. The diffusion of sensor technology from dedicated devices into everyday environments offers a potentially omnipresent, rich source of information that might be used by pervasive computing applications in multiple ways. An example of such an application is a constructive memory model extending the user's perception. With such a memory model, on the one hand context dependent support can be provided to the user by considering previous experiences in similar contexts. On the other hand, such a memory model could complement the user's natural memory and can be used to retrieve forgotten or unnoticed information at a later point. After some time in short-term memory, experiences are transferred to the long-term memory stage where they are linked with previous experiences. That way, general attitudes and preferences are established (often liking or disliking something without exactly knowing why), and experiences are related to similar ones, which helps to recall them later. Once again, both are features relevant for the ubiquitous application proposed in this thesis.

Context, especially when recorded over the long term, offers a wide range of possibilities to enhance the services provided by some computer system. These possibilities include inferring of current and past user actions, selection of devices and so forth. However, the prediction of future contexts based on the recorded past contexts is often conceived as the ultimate challenge in exploiting context histories. Context prediction, that is, inferring an expected future context, can offer distinct advantages over the sole usage of past and current contexts: Obviously, it can be used to perform actions on behalf of the user, but this is problematic. However, it is also possible to infer predicted context even without triggering actions in the physical world. On the one hand, comparing predicted contexts with recognised ones allows the system to detect irregularities and therefore assists in dealing with system failures. On the other hand, it can provide user interaction that conforms better to the user's expectations. The efficiency of interpersonal communication builds upon a shared understanding of the past, current and future contexts within which interactions take place. Computer systems usually do not share such an understanding, and therefore at least a partial awareness of the relevant contexts is a prerequisite for a significant improvement of user interaction.

Throughout the literature, there have been explanations and analyses of the concepts and structures of context awareness, memory and reasoning. There is a problem of knowledge engineering for context-aware systems and applications. The current default approaches have

put pressure on system designers with the obligation to anticipate all possible contexts, as well as determining the most suitable actions for each of these specific contexts. Anticipating all possible contexts is non-trivial at design time. This is due to some contexts being too complex to be defined and the large range of contextual information. The memory model of constructive memory is introduced as a way to help in overcoming these gaps and problems with context awareness. Integrating a constructive memory and reasoning component to context awareness will provide a way to learn and recall contexts, especially in the discovery stages of a system, from previous experiences. By incorporating a learning component as a part of the system, it has the potential of reducing design-time complexity by mapping all possible context environments with all possible contextual information. A constructive memory model can be computer-intensive and is not as good for small memory limited devices. To date, no such memory model has existed on mobile computing devices for context awareness. However, this research aims to develop a lightweight memory model for mobile devices to aid in the construction of contexts for context-aware applications. Table 2.1 provides a SWOT (strengths, weaknesses, opportunities and threats) analysis outlining the key areas taken from all of the literature presented in this chapter.

Chapter 3 presents the conceptual framework for this research.

Strengths	**Weaknesses**
o Combining sensed with non-sensed information aids in reasoning about context o Use of both static and dynamic contextual information helps with the accuracy of information being discovered o Use of both hierarchical and Cartesian models for describing location in context awareness assist in higher levels of interactions between users and applications o Context awareness in its current state, reduces the burden on users whilst creating software that is un-intrusive and *invisible* o Constructive memory assists in context prediction by recalling contexts from previous experiences where similar information is discovered	o Effects of erroneous, imprecise and ambiguous context information o Reliabilities of contextual information o Resource limitations in terms of memory storage and sensor failures which may record missing values o Accuracy in the type of sensor technologies being used o Modelling and training of sensor information discovered to a specific context is long and time consuming process o Applications lack any reasoning capabilities that facilitate in understanding a context o Context models fail to address uncertainties o Mobile context aware applications are both battery draining and memory intensive
Opportunities	**Threats**
o New improvements in context sensing technologies o Social and technological influences o Research developments in using context histories to predict contexts o New technology developments for mobile computing and memory storage o New environments in which mobile devices and applications operate	o Technological changes that make existing sensing technologies obsolete o Privacy issues relating to user acceptance o Security issues relating to contextual information being discovered o Robustness and sensitivity to operating environments o Social fears regarding personal freedom and information rights

Table 2.1. A SWOT analysis of the key areas taken from literature review.