A PRIVACY ENHANCING INFRASTRUCTURE FOR CONTEXT-AWARENESS

A THESIS SUBMITTED TO

THE UNIVERSITY OF KENT AT CANTERBURY

IN THE SUBJECT OF COMPUTER SCIENCE

FOR THE DEGREE

OF DOCTOR OF PHILOSOPHY

BY

PATRIK OSBAKK

DEDICATION

To my wife,

Without your love and support it had not been possible.

ACKNOWLEDGEMENTS

Firstly I would like to thank my supervisor Nick Ryan for his support and guidance. I am especially grateful for the freedom I have been given during my studies to explore the continuous stream of ideas I have had. Looking back I can see that many ideas have been on the periphery of my research and beyond. The experience gained from these explorations, successful and unsuccessful, is invaluable.

Many thanks also to David Shrimpton and Ian Utting, my supervisory panel, for their help and suggestions over the past years. The friendly discussions we have had, often over a cup of coffee, have undoubtedly contributed to pushing my research forwards as well as providing essential feedback. Special thanks also to Janet Linington for your support over the years and for setting in motion my career with Java.

I would also like to thank the Computing Laboratory including the Applied and Interdisciplinary Informatics group, the Networks and Distributed Systems group, the academic staff, my colleagues, the system administrators, and last but not least the administrative personnel who have all aided me in my research and made my time at the University of Kent enjoyable. I am also very grateful for having had the privilege of receiving an EBS Home Bursary and a Maintenance Award, allowing me to pursue my dreams.

I would also like to acknowledge the backing provided by my employer in accommodating my needs to complete this thesis.

Finally, I would like to thank my family and friends for their support and understanding during these demanding years. That you have stood fast whilst I have had to prioritise studies and work means a lot to me. I am grateful always.

TABLE OF CONTENTS

FIGURES

ABSTRACT

Context-awareness enables applications and services to better fulfil the needs of users by adapting to their situation and their preferences. However, the use of contextual information is complicated by privacy concerns. A subject's context is personal and needs to be regarded as sensitive. Hence, contextual information must only be used with the consensus of the subject and according to their privacy preferences.

This thesis examines the development of privacy-friendly context-aware systems. In particular the focus is on (A) improving the overall level of privacy, (B) evaluating access control mechanisms, (C) providing development support, and (D) offering protection to third-party infrastructures. The hypothesis investigated is whether these objectives can be achieved through the use of a privacy enhancing infrastructure.

As part of the investigation two conceptual models are presented describing the assumptions made about context and privacy. Also presented is a decentralised privacy enhancing infrastructure developed and implemented to determine the validity of the hypothesis. Along with the infrastructure mechanisms for privacy protection including authentication, access control, and anonymity are discussed. A general data format for context communication in the infrastructure is also presented.

Finally the thesis presents the findings uncovered during the investigation and evaluation of the hypothesis. This includes a qualitative analysis of whether the privacy enhancing infrastructure meets the key objectives, a user survey examining the performance of two candidate access control mechanism, a performance measure of the infrastructure when run with resource constrained devices, and a comparison with the approaches taken in related work.

CHAPTER 1

BACKGROUND

The vision of ubiquitous computing is no longer pursued only by a handful of people. It is now actively being researched on a much larger scale. Various prototype systems have been developed and deployed that provide a wide range of services in many different environments.

This chapter introduces the area on which this thesis focuses, namely privacy in context-aware ubiquitous computing. It outlines and discusses some of the previous work done in the fields of ubiquitous computing, context-awareness, and privacy. The intention is to give a broad overview to the fields and to present the work that has inspired further research. Each section will try and capture a different aspect of the research area.

1.1. Introduction

Context-awareness has the potential of providing significant benefits. By utilising contextual information applications and services can adjust to our situation and preferences, thus enabling new functionality and an improved user experience to be provided. For example, context-awareness enables a city guide application to not only list restaurants but also make recommendations on places that are both nearby and that serve food to your taste. Influenced by the vision of ubiquitous computing, these solutions are to be pervasive, providing unobtrusive services when and where appropriate.

Whilst beneficial, the use of contextual information is not without controversy. Privacy is of great concern in context-aware systems. Contextual information is by nature personal and sensitive and when collected over time it reveals an individual's behaviour and preferences. This can be exploited by dishonest individuals. Given the existing misuse of personal information on the Internet, it

is believed these concerns are well founded. Hence, contextual information must be carefully handled and protected.

Consequently, the work presented in this thesis focuses on the area of privacy protection and context-awareness. In particular, the thesis examines an infrastructure approach to providing development support for privacy-friendly context-aware systems.

1.2. Ubiquitous computing

The concept of ubiquitous computing was first thought of at the Electronics and Imaging Laboratory at Xerox Palo Alto Research Centre (PARC) in late 1987 [Weiser, Gold et al.1999]. It has since gathered momentum and is now widely researched. This section will summarise the initial work done at Xerox PARC.

The original proposal leading to the birth of ubiquitous computing was for the creation of wall-size computer displays. These displays intended to be capable of providing both data output and input, yet be as easy to use as whiteboards. This idea inspired the new vision of computing, where computers are invisibly spread throughout our environment. Another contributing factor was the work done by anthropologists in the area of Work Practices and Technology at PARC. Their studies, into how people use computers, led M. Weiser and others to think more about the situations in which technology is used rather than on technology alone. Together this gave birth to the Ubiquitous Computing program in the Computer Science Laboratory (CSL) at PARC.

The initial goal of this program was to solve some of the problems they felt personal computers exhibit including that computers are too hard to use, they need too much attention, and they isolate us from other people [Weiser, Gold et al.1999]. Although on first inspection these appear to be issues regarding the graphical user interface, it was apparent at PARC that it was the result of the whole machine and its usage [Weiser 1993]. To quote M. Weiser “The challenge is to create a new kind of relationship of people to computers, one in which the computer would have to take the lead in becoming vastly better at getting out of the way, allowing people to just go about their lives” [Weiser 1993]. Their answer to this challenge was the development of three new

computing devices known as tabs, pads, and boards. Weiser describes tabs as inch scale devices approximating active post-it notes, pads as foot scale devices behaving like notepads, and boards as yard scale devices as the equivalent to whiteboards [Weiser 2002]. Examples of these new types of devices are the ParcTab, ParcPad, and LiveBoard.

Although the initial goal of the Ubiquitous Computing program was thought of as an answer to the many problems with personal computers, the contributions that this research program has made are far more important. It resulted in the creation of a completely new research area and defined ubiquitous computing as we know it.

1.3. The Active Badge System

One type of context that has been widely researched is location. An early and widely know example of a context-aware, more specifically location-aware, system is the Active Badge system [Harter, Hopper 1994] developed at Olivetti Research Limited, later AT&T Laboratories Cambridge.

An active badge is a small wireless device that transmits a unique identifier at specified intervals over infrared. It is a square device about 55 x 55 x 7mm with a weight of 40g [Want, Hopper 1992] and it has two buttons, two leds, and a speaker [Harter, Hopper 1994]. The active badge location system uses infrared as well as radio technology to sense the location of these badges, which can be worn by people or attached to objects. The system works by having a fixed network of infrared sensors that are positioned at known locations. This network of sensors constantly listens for incoming signals from active badges. When an active badge is within range of a sensor the signal it transmits will be received and since the location of the sensor is known, the system can establish the identified badge's location and thus also the associated person or object. Active badges can with this technique provide "room scale location" [Harter, Hopper 1994 (p.2)].

Better granularity of the location is described to be achievable by using a denser network of sensors and reducing the transmitter power of the badges. Harter and Hopper also describe another hybrid infrared radio technique employed by the

Active Badge system to improve the location granularity in certain areas [Harter, Hopper 1994]. This works by embedding a passive radio receiver into the badges and by placing radio transmitters around objects of interest. The transmitter sends out an identifiable signal that is directed such that it forms a field around the object. A badge entering the field then adds the received signal to the transmission it sends to the infrared sensors. Thus the badge can be located to be within the field, achieving what they call "desk scale location" [Harter, Hopper 1994 (p.3)].

The initial application developed for the active badge system was intended to be used by the receptionist at Olivetti Research Limited to help with the forwarding of telephone calls [Want, Hopper 1992]. The application displays a list of people's names along with their last known location and the telephone extension there. It also displays a measure of how accurate the location information is or how old the information is if data is not being updated. The system also handles the processing of simple user commands. Among the interesting ones described are WITH which lists the badges near a supplied badge name, LOOK which lists the badges near a supplied location, and HISTORY which produces a report containing information about the whereabouts of a supplied badge name during the last hour.

It is stated that over 1500 badges and 2000 sensors have been deployed (November 1993) [Harter, Hopper 1994] and that the system has been accepted and is used daily by the staff at Olivetti Research Limited [Want, Hopper 1992]. As such the active badge system has demonstrated the feasibility of deploying location sensing in working environments. However it is also described that despite its successful internal deployment, people from outside are sceptical about whether they would like such a system in their office due to privacy concerns. Hence privacy is an issue that most definitely needs to be addressed and even more so when considering more accurate location systems like the successor of the active badge, the bat system [Addlesee, Curwen 2001].

1.4. The Context Toolkit

The work on context-awareness done at Georgia Institute of Technology focused on improving the development support for context aware applications through the development of a conceptual framework. This section will present some of this work starting with their definitions of context and context-aware.

The most widely accepted definition of context is probably the one presented by Dey and Abowd, which states that: "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves" [Dey, Abowd 2000A]. In addition to defining context they provide a system for categorising different types of context [Dey, Abowd 2000A]. The system consists of two levels. On the first level are what they consider to be primary pieces of context: location, identity, time, and activity. These are the important pieces of context that characterise a situation. On the second level they place all other types of context, which are considered to be secondary. Secondary pieces of context are attributes of entities with primary context and can be indexed by primary pieces in information spaces. Dey and Abowd continue by providing a definition for context-aware which state that "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" [Dey, Abowd 2000A]. They also propose a categorisation of the features context-aware applications have. Features are categorised into one of three categories: presentation, automatic execution, and tagging. Where the presentation category includes actions such as presentation of context information or services to a user, the automatic execution category includes actions that are triggered or adapted by the context of a user, and finally the tagging category includes actions that add context to information for later use.

In his thesis, *Providing Architectural Support for Building Context-Aware Applications*, [Dey 2000B] Dey points out that many of the well known context-aware applications only support a small subset of types and features. Dey

attributes this lack in range to the difficulty of using context information. To make the development of context-aware applications easier a framework is proposed, allowing developers to focus on the problem at hand. Seven features are described as being required in the framework:

1. The framework must enable an application developer to specify the context required by an application.
2. The processing of context should be independent of the retrieval.
3. Contextual information should be able to be interpreted transparently by the framework.
4. The communication between the context source and the applications should be transparent and support distributed sources.
5. The context sources should be constantly available to allow applications to retrieve contextual information when needed.
6. The framework should store captured contextual data to allow access to a context history.
7. The resource discovery should be built into the framework.

It is recognised that not all solutions can be accommodated. As such the framework provides four basic building blocks which can be used to extend the support: widgets, aggregators, interpreters, and services. The widgets act as context sources. They provide the necessary abstraction from the physical sensor. Aggregators allow the contextual information from multiple widgets to be combined to reduce the complexity of having to communicate with the widgets individually. Interpreters are used to provide high-level context information from the collected low-level information. Finally services provide an output, allowing changes to be made in the environment. With these building blocks Dey wants to encourage the development of new components for the framework rather than standalone ad hoc solutions.

The implementation of the framework described in Dey's thesis [Dey 2000B] is known as the Context Toolkit [Dey, Abowd 2000C]. The toolkit is implemented using Java and provides the required features of the framework [Dey 2000B (s.4.2.2)]. Communication between the components uses the HTTP protocol and the data is encoded using XML. They also emphasise that each component runs

independently allowing them to be used by many applications. To demonstrate the benefits provided by the toolkit Salber and others [Salber, Dey et al. 1999] describe three applications that have been built: an in/out board, an information display, and a meeting board. The in/out board displays whether or not a person is in the office building. To do this the board collects and uses the entry and exit time of office members. They also describe an information display that shows information relevant to a nearby person. This display uses context information like the identity of the person, what group they are associated with, and the location of the display. Finally they have added context-awareness to DUMMBO, another project at Georgia Institute of Technology [Brotherton, Abowd et al. 1999]. DUMMBO is a smart whiteboard that captures what is written on it during meetings as well as what is said. This information is then made available afterwards. By utilising the Context Toolkit they altered the starting behaviour of the whiteboard so that it started recording in the presence of two or more persons.

The work done at Georgia Institute of Technology provides a good base for further research. First of all their definitions of context and context-awareness allow for a better understanding of what context-aware systems are. Secondly their work on a framework and toolkit that aid the development of context-aware applications have been an important start to the move away from ad hoc implementations towards more uniform structures. However one area in which further research is required is that of privacy. Although a preliminary extension is presented [Dey 2000B (s.6.1)], allowing some control over the access of information, privacy protection is not central to the framework design. The nature of privacy issues requires this to ensure adequate protection is provided. Also since the development of the Context Toolkit there has been a movement towards infrastructures being requested that can provide even greater abstraction for developers [Hong, Landay 2001].

1.5. Cooltown

In the Cooltown project at Hewlett-Packard Laboratories it is believed “that the future consists of nomadic people carrying personal communication and web browsing devices interacting with services that are location specific and

customized to the user" [Debaty, Caswell 2000]. The project seeks to support this behaviour by adopting the existing web infrastructure such that the real and virtual worlds are brought closer together. At the heart of the project there are three different categories of entities: people, places, and things [Debaty, Caswell 2000]. People are the users in Cooltown, a place is an area or space, and things are objects. As such people can use things, and places can be filled with both people and things. The vision they have is that each one of these entities will have a web presence [Debaty, Caswell 2000], i.e. it needs to have a representation on the web that can be accessed using an URL. This web presence is then used to provide customised and enhanced services to people.

The Cooltown museum is one of the examples they give of what an enhanced service may look like. Throughout the museum beacons have been deployed close to objects of interests such as pictures. These beacons transmit URLs that link the physical objects to their respective web presence, where information about the object is found. The visitors can then receive these URLs with their portable digital assistants (PDA) when they explore the museum, allowing them e.g. to read more about the object using the PDA's web browser. The URLs can also be stored for later use. In the museum bookshop URLs are associated with the items for sale, to allow visitors to retrieve further information such as reviews. Furthermore, they offer a print service that makes it possible for visitors to print out the web pages associated with the URLs collected during their stay, as well as reproductions of paintings.

The future as thought of in the Cooltown project is not far away. Today a large proportion of the population in the industrial world carry mobile phones, some of which support web browsing, and the use of PDAs, and increasingly, smartphones is steadily growing [Debaty, Goddi et al. 2003]. These mainstream devices can already today be utilised to gain access to enhanced services that give you more information about museum artefacts [Duan 2002], help you find your friends [Telia 2005], or guide you around a tourist attraction [Mobil Turism 2005]. But there is still some way to go before the real and virtual worlds seamlessly interact; simply providing a web presence for people, places, and things is not enough.

1.6. Music FX

Another project that shows some practical uses of context information is MusicFX [McCarthy 1998]. It investigates the use of a group preference agent in a shared environment. The environment in question is that of a fitness centre, the Fitness Xchange at the Accenture Technology Park in this case. In this fitness centre, as in others, music is often being played. The type played can often be subject to dissatisfaction, and this is indeed what 25% of the Fitness Xchange's written suggestions convey. This project has therefore developed a system that aims to improve on this by adjusting the type of music played according to the preferences of the group of people working out.

To be able to make a decision about what music should be played MusicFX has a database containing people's preferences with respect to musical genre. The preferences are expressed as a collection of ratings for the different genres. Each rating is on a 5 step scale, going from love +2 to hate -2 with 0 being the no preference. This database is populated by the members and can be updated at anytime. The system must also know which members are present in the environment. This is achieved by requiring people to logon to the system when they come to workout. After logging on that person is by default assumed to be present for 90 minutes, removing the need to explicitly logout. Together these pieces of data form the basis needed for MusicFX to run the group preference arbitration algorithm, which is the decision maker. The algorithm is run every time certain events occur, e.g. when a member logs on. Without going into too much detail the algorithm calculates the aggregate rating of the available genres using the preferences of the current people logged in. The top rated genres are then short-listed and given different weights that reflect the aggregate ratings. It is then from this list a genre is selected; the selection is random but takes into account the different weights. Having selected a genre to play MusicFX utilises an existing music service to play the appropriate music. This service has previously been controlled manually by the members of staff and consist of 91 channels each catering for a certain genre. In addition to this there are a number of ways in which the system can be fine-tuned to ensure there is variation in the music played and for users to express their dislike of songs.

The MusicFX project shows one way in which context-awareness can improve our environments. It is especially interesting, seeing as music is commonly played in the background of places such as shops and restaurants and it is also something people can easily relate to. However the project does not consider the scenario of users moving between different environments nor does it adequately address the privacy issues that arise with the introduction of such a system. The set of preferences used by MusicFX are stored locally in a database, as such data will be duplicated if other services requiring a user's musical preferences are deployed. It also gives the user less control over the use of their contextual information and its accuracy. Furthermore by storing preferences when not needed, i.e. between workout sessions, the system raises questions regarding the extent to which the preferences are used for other purposes than influencing the type of music being played.

1.7. Fieldwork

The use of context-awareness extends beyond controlled environments such as offices, homes, and gyms. It can also be applied to other situations, for example to assist in fieldwork. The Mobile Computing in a Fieldwork Environment (MCFE) project is one effort aimed at developing context-aware applications for the field [Ryan, Pascoe et al. 1997]. The project has developed several tools for handheld devices that support the recording, presentation, and administration of field notes [Ryan, Pascoe et al. 1997] [Pascoe, Morse et al. 1998] [Ryan, Pascoe et al. 1999].

Among the tools developed is a general purpose system for rapid data entry [Pascoe, Morse et al. 1998]. The system is based on the stick-e note concept [Brown, Bovey et al. 1997], but is distinct in that the focus is on recording information instead of retrieving information [Pascoe, Morse et al. 1998]. To facilitate the fieldworker's need for quick note taking the system utilises both templates and automatic capture of contextual information. The prototype developed has been tested in field trials in Kenya, where animal behaviour was studied. It was concluded from the trial that the tool developed in addition to replacing traditional pen and paper also benefited the users in offering better

usability and speed [Pascoe, Morse et al. 1998]. Thus, the work shows that context-aware tools are both applicable and useful in field work environments.

Another tool developed as part to the MCFE project is a geographical information system allowing the mapping of field notes on handheld devices [Ryan, Pascoe et al. 1999]. The system utilises contextual information to make note taking easier, automatically augmenting notes with positioning information, date and time, and the recorder's name. Previously recorded information as well as the user's current position is available using a map view. From the map field notes can be retrieved for viewing and editing. Field trials with this tool have also verified the applicability of context-aware mobile systems to fieldwork [Ryan, Pascoe et al. 1998]. Its usefulness has also prompted continued development of the tool [Ryan 2005].

What these applications have demonstrated is that mobile computing and context-awareness can be successfully applied to fieldwork. Hence, it is important to not limit the focus of context-aware work to areas such as offices and homes, but also consider a wider range of environments.

1.8. Privacy and Freedom

The book *Privacy and Freedom* by Alan F. Westin [Westin 1970] provides a ground for understanding privacy by addressing some of the important questions. It is also the source of a widely used definition of privacy.

So what is privacy? According to Westin "Privacy is the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others" [Westin 1970 (p.7)]. With respect to the interaction between an individual and society he states that "privacy is the voluntary and temporary withdrawal of a person from general society through physical or psychological means" [Westin 1970 (p.7)]. The book also presents four basic states of privacy: solitude, intimacy, anonymity, and reserve. Solitude is the first and most private state. In this state the individual is not part of any group or under any observation. The only disturbance that can be experienced originates from the individual's own senses and psychological fears. The second state is intimacy. In this state the individual

is part of a small group that is allowed to be separate from the rest of society. The group members can therefore form an open relationship. The third state is anonymity. In this state privacy is attained by being in a larger public group of people. The individual may be observed but since they are anonymous in the group they may still enjoy privacy. The fourth and final state is that of reserve. In this state psychological barriers are used to protect privacy. Such barriers may involve limiting the information an individual communicates about themselves.

Is this a modern concept? No, it is argued in *Privacy and Freedom* that the need and desire for privacy is not a new phenomenon but that it can be traced as far back as to our animal origin. To support this, a number of parallels are presented between humans and animals. For example, Westin states that almost all animals, or intimate groups of animals, seek periods of seclusion. It is also said that animals, just as humans, possess mechanisms that govern the distance between members of a group, allowing personal space. Related to this is the requirement for animals to have a minimum amount of private space to ensure health and survival. Furthermore it is said that animals share our need for intra-species social interaction. As such privacy is not only a human desire or as Westin puts it "the quest for privacy is not restricted to man alone, but arises in the biological and social process of all life" [Westin 1970 (p.11)].

What function does privacy play in society? In *Privacy and Freedom* the functions of privacy are discussed with respect to democratic societies and four different categories are described for which privacy is essential: personal autonomy, emotional release, self-evaluation, and limited and protected communication. Privacy is needed to ensure personal autonomy because the most serious threat to an individual's personal autonomy is, according to Westin, that someone may learn their ultimate secrets, be it deliberately or by accident. Then there is emotional release. Emotional release can be attained by, for example, the temporary discard of social roles, by not fully complying with social norms, or by being allowed to express anger. All of these forms rely on the individual not being held accountable for their release actions, thus requiring privacy. The third category described is self-evaluation. Individuals need to be

able to evaluate collected information about themselves and others, and they need to do so in privacy. One example given is that although individuals consider the morality of their actions continuously, it is in privacy they compare it to their personal ideals. Finally there is privacy for limited and protected communication for which Westin describes two general aspects. First, it serves to give the opportunities required for an individual to share sensitive information with those they trust. Secondly, it allows individuals to set psychological boundaries with respect to their interaction with others. Finally it is important to note that even though privacy fulfils an important function in society, "the individual's desire for privacy is never absolute, since participation in society is an equally powerful desire" [Westin 1970 (p.7)].

So what determines the balance of privacy? In *Privacy and Freedom* the political system is described as a "fundamental force in shaping its balance of privacy" [Westin 1970 (p.23)] since the requirements differ from system to system. As an example, the contrast between a totalitarian state and a democratic society is presented. In a totalitarian state privacy, or if you will secrecy, is restricted to the regime. All others are subject to high surveillance and disclosure. On the other hand in a democratic society it is the government that is under the scrutiny of the public and privacy is used to protect the private life of the people. Furthermore historical and political traditions as well as cultural differences are also recognized as affecting the balance of privacy. But it is not only society that determines the balance of privacy. An individual's personal status, their life situation, as well as their personal preferences affect this balance too. Finally Westin emphasises that an individual will constantly adjust their balance of privacy to allow them to both fulfil their role in society as well as their personal needs.

Although *Privacy and Freedom* was published 1970 the initial chapters are still relevant in today's information age. We still desire privacy [Guardian 2002] and the basis on which society operates is fundamentally the same. It is therefore assumed that privacy still fulfils much the same function today. The major differences are that the quantity of information flowing is larger and the speed at which it is communicated much faster. At the same time the amount of human

intervention needed to collect and distribute information has decreased, resulting in a greater potential for privacy invasion.

1.9. P3P

The Platform for Privacy Preferences Project (P3P) [World Wide Web Consortium 2002A] is a specification, endorsed by the World Wide Web Consortium (W3C). With the specification, W3C aims to achieve two goals. Firstly they want to enable websites to specify their privacy policy/policies in a standard machine-readable format. Secondly they want users to be informed about what data is collected, why it is collected, and if possible what the user can opt-in or opt-out for. In other words, P3P is intended as a mechanism with which sites can convey their privacy practises and their intentions to users and user-agents.

The P3P specification [World Wide Web Consortium 2002A] describes a format in which P3P policies, i.e. privacy policies, can be represented. The format is XML-based and is accompanied by a standardised vocabulary. If required this vocabulary can be extended. The specification also includes information on how to reference P3P policies on websites and the meaning of the accompanying vocabulary. What the specification does not cover is how to enforce or verify the P3P policies described. The W3C also publish a related specification that outlines how to represent privacy preferences with respect to P3P policies using sets of rules [World Wide Web Consortium 2002B]. The format described by this specification is also based on XML, and features an extendable vocabulary.

So how will P3P be used? The specification [World Wide Web Consortium 2002A] outlines a simple example describing the use of P3P in a Web environment. A user desires to visit a website on the Web and at their disposal is a web-browser with a built in P3P user-agent. The user types in the address for the desired webpage and at this point rather than directly requesting the page the P3P user-agent will first attempt to retrieve the site's P3P policy. Assuming a policy is retrieved, the user-agent will then evaluate the policy to see if it matches the user's privacy preferences which have been provided earlier. If the policy matches the preferences the page will then be retrieved and displayed. If

the policy does not match, the user may be prompted for a decision on whether to retrieve the page or not. Alternatively the agent can be set up, for example, to block the page automatically.

Another use is described in the paper *Privacy Enhancements in the Mobile Internet* [Nilsson, Lindskog et al 2001]. There Mikael Nilsson et al. describe how P3P can be used to protect Composite Capabilities/Preference Profiles, a standard for describing device capabilities and user preferences [World Wide Web Consortium 2004E]. The approach they take requires a device/user to have a minimal profile in addition to their normal CC/PP profile. The minimal profile may be empty or it may contain data that is not considered sensitive. As such it can be used before a level of trust has been established between user and the site. Hence, this minimal profile is used when the user-agent requests a site's P3P policy. The P3P policy is then evaluated to see if the site's privacy policy matches the user's requirements. If it does then the more elaborate CC/PP profile will be used for the subsequent requests. If the site's policy does not fulfil the user's requirements they suggest the continued use of the minimal profile. They also suggest that the minimal profile can be used to access non-P3P compliant sites.

The example above demonstrates how P3P can be used. Whether P3P does indeed improve the privacy of users or not is debated widely. In the report *Pretty Poor Privacy: An Assessment of P3P and Internet Privacy* [EPIC, Junkbusters 2000] the Electronic Privacy Information Center (EPIC) and Junkbusters raise their concerns regarding P3P. They are for example worried that users will be forced to give up privacy or accept a more restricted Internet. Another issue they highlight is that there is a problem with launching P3P because if few sites use P3P then few users will invest time in setting it up and then there is no incentive for sites to use P3P. Concerns are also held due to the fact that it is not possible to make sure that a site follows its stated P3P policy. Several other issues are also discussed and the overall view presented in the report is that P3P will actually not improve user's privacy.

In the paper *Can P3P Help to Protect Privacy Worldwide?* [Grimm, Rossnagel 2000] by Rüdiger Grimm and Alexander Rossnagel a slightly more optimistic

view of P3P is presented though. They show that P3P can complement the German privacy laws and that it supports some of the legal requirements sites have to fulfil. They also point out that P3P could be extended to better support the existing legal framework. Another positive effect mentioned is that P3P can increase the awareness of users regarding privacy protection. However, several issues with P3P are also highlighted in this paper. For example, they criticise the limited choice the user has, i.e. either to accept or reject a policy. They point out that there is no way for the provider to know why users reject a policy. They also feel that the lack of technology to ensure policies are enforced is a problem, especially where a self-regulatory approach to privacy protection is used. However, it is concluded overall that the use of P3P can be beneficial.

Independent of what view you hold, the development of the Platform for Privacy Preferences (P3P) specification must be credited for fuelling the debate regarding internet privacy. Raising the awareness among both developers and users about internet-privacy is just as important as providing technological solutions, if not more so.

1.10. Summary

This chapter has presented a selection of the previous work done in the fields of ubiquitous computing, context-awareness, and privacy.

The initial research on ubiquitous computing done at PARC still remains an important influence on today's research. Developing systems that are all around, yet invisible, is increasingly desirable. The ongoing technological development is turning this vision into reality by allowing smaller and smaller devices to be manufactured. Also, the everyday use of technology like mobile phones will gradually make the interaction with ubiquitous devices more transparent to the user. Research projects such as Cooltown, The Guide project, and FieldWork have shown that there are practical uses for ubiquitous computing technology in different areas.

The progress made within ubiquitous computing has also allowed another research field to be born, that of context-aware computing. Dey et. al. define a context-aware system that uses contextual information to provide a user with

information or services relevant to their task. One piece of context that has been widely researched is location. The Active Badge system, for instance, demonstrated how location information can be captured indoors using infrared and radio based sensing technology and that the deployment of such technology is feasible. But context-aware applications are not limited to location. MusicFX, for example, uses the gym members' preferences when determining what music to play in the gym. As such, context-awareness can benefit users in many situations. However context-aware computing also brings its own set of challenges. The work done on the Context Toolkit at the Georgia Institute of Technology emphasises the difficulty in using context information and the need to provide support for the development of context-aware applications. Also, context-aware applications operate in a wide range of environments. The projects described in this section cover environments such as the office, the gym as well as more general fieldwork environments. This undoubtedly provides issues itself.

Privacy is a long standing desire that according to Westin can be traced back as far as our animal origin. It also asserted that privacy fulfils important functions in society by, for example, providing people with the opportunity to evaluate gathered information. It is thus not surprising that privacy is a subject of concern to ubiquitous technology. The Platform for Privacy Preferences Project (P3P) is one attempt in improving the privacy of online technology users. Although developed for use in traditional environments, it has been shown that the technology can be used within mobile environments as well. How successful P3P is in improving privacy has been the subject of controversy, and is likely to continue to be so. The importance of this technology is instead seen to be the awareness and discussion it has raised regarding online privacy.

Although each of these fields are research areas in their own right, the existence of a strong link is apparent. To provide valuable and acceptable ubiquitous services to end users it is believed that we need to consider all three of these research areas. In the next chapter a project will be introduced that attempts this, i.e. combines the fields of privacy, context-awareness, and ubiquitous computing.

CHAPTER 2

THE PROJECT

As the previous chapter has shown the development of ubiquitous computing and the related field of context-awareness have made some significant progress. A multitude of research projects have been undertaken which further the understanding of these types of system. The work done has also highlighted a number of existing issues with ubiquitous and context-aware computing and emphasised the need for further work to address these.

This chapter presents the motivation for the research undertaken that is presented in this thesis within the field of Privacy in Context-Aware Ubiquitous Computing. It also describes in which ways this research aims to contribute to the field, along with the issues that need to be addressed. Finally the chapter also present some of the related work undertaken in the field, in the context of this project.

2.1. Motivation

Over the last decade or so, significant progress has been made in the manufacture of small devices. Small computing devices are no longer limited to research or corporate use, they are available to mainstream users at what can be considered reasonable cost. They have also become more powerful, and even though resources are still limited, what they can do has increased and continues to do so. Devices may still be far from invisible but even so a notable step has been taken towards the wide scale realisation of ubiquitous computing.

Although this development itself has been a source for inspiration, the motivation for this work concerns the issues that exist in developing publicly acceptable context-aware ubiquitous systems for widescale deployment. In

particular the motivation comes from the work needed in two key areas: privacy and development support.

2.1.1. Privacy

The distribution of invisible computer devices, which are able to communicate and store information, throughout our physical environment poses some serious questions about personal privacy. What devices are nearby and what purpose do they serve? Do they collect information? If they do, what type of information is collected and will it be distributed? The origin of these concerns are in fact the very desirable characteristics of ubiquitous computing, the ubiquity and invisibility. Although privacy concerns were encountered already within the Ubiquitous Computing program at Xerox PARC [Weiser, Gold et al. 1999], these issues have not been prioritised. This has left privacy being recognised but to a large extent unaddressed.

Furthermore the development of context-aware applications can be seen to aggravate any existing privacy concerns held with respect to ubiquitous computing. The nature of context information is the cause of these heightened concerns. A piece of context can consist of anything from location information to that of the current activity. Thus the information will often be personal and considered to be sensitive. Moreover if one takes into account that a context-aware application may collect and use many pieces of information of various types over an extended period of time, then it is not difficult to see that there will be an impact on people's privacy. Brown and Jones go so far as to state that "Context-aware applications, above all others in the pervasive field, can be regarded as anti-privacy" [Brown, Jones 2004].

But are people really concerned about their online privacy? Surveys examining peoples' feelings towards privacy suggest that they are. For instance in the GVU's 10th WWW survey from 1998, see Figure 1, 52.8% of the respondents said that they were in general very concerned about the security on the Internet, where security was stated to include issues like privacy, confidentiality, and authentication [GVU's WWW Surveying Team 1998].

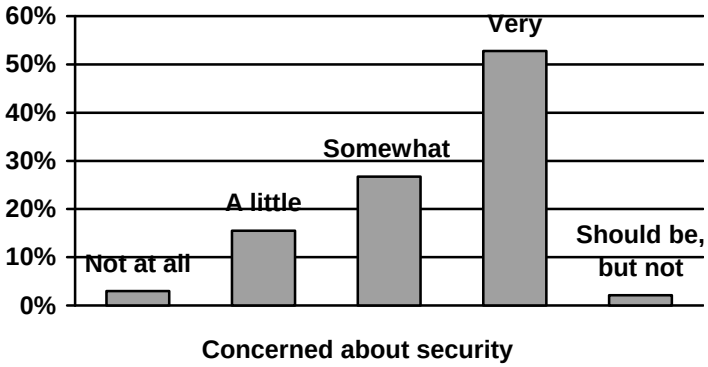


Figure 1. Concerned about security? [GVU's WWW Surveying Team 1998]

Similarly an ICM Poll published by *The Guardian* in 2002, see Figure 2, shows that 66% of those asked agreed to the statement: "I am worried about the security of my personal information travelling on the internet and email" [Guardian 2002 (p.3)].

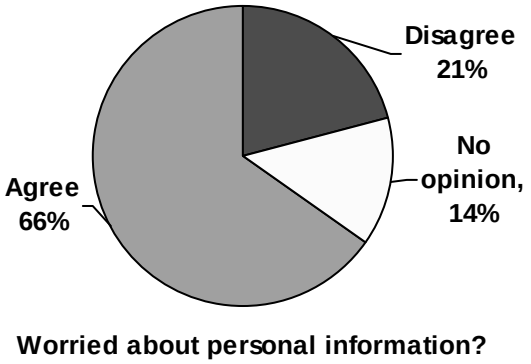


Figure 2. Worried about personal information? [Guardian 2002 (p.3)]

Finally a survey undertaken by *Your voice in Europe*, see Figure 3, shows that 69% of the respondents were afraid that the personal data they provide whilst buying or using online services will be misused [European Commission 2002]. In fact it was stated to be one of the main reasons why purchases were not made online.

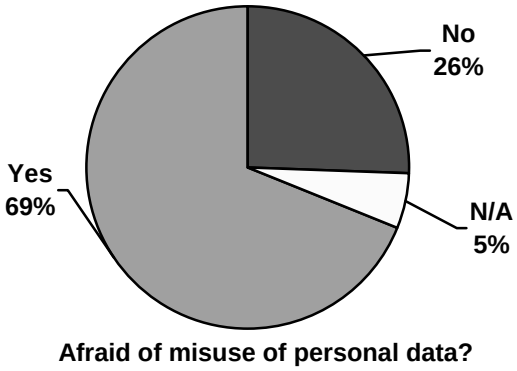


Figure 3. Afraid of misuse of personal data? [European Commission 2002]

All of the above surveys focus on slightly different aspects but they all show a clear trend. Online privacy, security, and trust are important as well as desired.

With surveys clearly showing that privacy is indeed desired and context-aware ubiquitous computing having matured, privacy research in this field has become a hot topic. Privacy is indeed one of the key issues that are highlighted in the ubiquitous computing grand challenge [Crowcroft 2003]. And as progress is made in creating truly ubiquitous and context-aware computing environments, we will increasingly find privacy to be the subject of concern. The need to address these concerns is further stressed by the desire to deploy context-aware ubiquitous systems on a wider scale for which public acceptability is ever so important.

The development of privacy protection is therefore a key motivation for the work presented in this thesis. Most would agree that the thought of a world where information about us is recorded, used, or even distributed without our knowledge is not very pleasant and that every effort must be made to avoid the creation of a surveillance society.

2.1.2. Development support

The vast majority of people are not expected to start using context-aware ubiquitous technology unless it provides them with some form of benefits. Thus the existence of useful applications will be very important when it comes to persuading people to use context-aware applications. It is also recognised that what constitutes a useful application will vary greatly from user to user. Hence

there exists a need to develop a wide range of applications, providing users with a choice.

But just as with the development of any other end-user application a significant effort is required in producing useful context-aware applications. The task is further complicated because of the nature of context-awareness. For instance four aspects described by Dey [Dey 2000B] as making the use of context information difficult are that:

- Limited experience exists in using context sensing devices.
- Abstraction is needed to make effective use of sensed context.
- Multiple dissimilar sensors may need to be used.
- Context information is dynamic.

Add to this list the need to develop secure applications that are privacy-friendly, then further difficulties arise including: the provision of appropriate access controls, integrity and secrecy of communication, secure data storage, etc. Hence the complexity becomes even greater.

It is therefore essential that support is provided for the development of context-aware application. By moving the responsibility for common tasks away from the application, the complexity can be reduced. This will allow the applications to focus on the task at hand and be developed with greater ease. Projects, such as the Context Toolkit [Dey, Abowd 2000C], have indeed shown this to be true. With further work in this area it is expected that application development can be made even easier. Others [Hong, Landay 2001] have indeed argued that there are benefits in taking an infrastructure approach rather than using a toolkit. Also the additional difficulties that appear when secure and privacy-friendly applications are developed creates a need for further work in this area.

Thus another key motivation for the work presented in this thesis is the need to study further how to provide better development support.

2.2. Focus

The scope of possible work within the field of study is large, even within the key areas of motivation. Indeed both context-awareness and privacy are presented as unsolved key issues in the ubiquitous computing grand challenge [Crowcroft 2003]. Thus the focus of the work has been further narrowed.

Firstly, the nature of the research has mostly been applied. By applied it is meant that the work have been focused on researching practical and operational solutions rather than abstract concepts. This follows the path of previous work in the field of ubiquitous computing where the emphasis has been on experimental research. Also, because the field of study is a combination of three independent fields, i.e. privacy, context-awareness, and ubiquitous computing, many benefits can be achieved by furthering the research into their integration.

Secondly, rapid progress is made in both the manufacture of small devices and in the available software for such devices. This means the area is a moving target. To avoid the research becoming a constant quest in utilising newer technology the work has focused on using the same set of devices and software as much as possible. Naturally newer technology has been used when appropriate, provided no significant overhead is caused. Although it is appreciated that benefits can be had from always using the latest available technology, it is seldom required.

Finally, the work has focused on researching the software side of ubiquitous computing rather than the development of specialised devices. Although the research into hardware is both useful and interesting it requires a large commitment of resources and time. Given the availability of small general purpose devices, e.g. PDAs, it is thought that the current focus will better utilise the resources to hand. Thus the work has used off-the-shelf devices throughout. Although this inevitably imposes limitations on what can be done, these are considered to be negligible. It should also be noted that the use of standard equipment greatly simplifies any large scale deployment of context-aware applications as equipment already in the hands of the users can potentially be used.

2.3. Intentions

The work presented in this thesis aims to contribute to the ongoing development in the combined field of privacy, context-awareness, and ubiquitous computing. In particular the intention is to contribute to the following four areas:

- The improvement of a user's overall level of privacy.
- The evaluation of different access control mechanisms.
- The provision of support for easy and rapid development of privacy-friendly applications.
- The protection of third-party infrastructures.

The first two areas focus on the need for privacy protection in context-aware ubiquitous systems, which is the primary motivation for the work. It is firmly believed that users should not need to compromise their privacy to be able to benefit from this technology. The intention is therefore to show that by employing careful privacy-aware design and by providing users with greater control over the distribution of their personal information, privacy can be maintained. The work also intends to show that a suitable access control mechanism is essential in providing the control required and includes an evaluation of different access control mechanism with respect to context-awareness.

The third area focuses on providing development support, another key motivation. The intention is to demonstrate that by using a privacy-enhancing infrastructure the handling of context information can be moved away from the application. It is also the intention to show that by doing this the design of privacy-friendly applications will be made both easier and quicker.

Finally, the last area on which the work focuses is the provision of privacy protection for existing context-aware systems. The intention is to show that by integrating the proposed infrastructure with other context-aware systems, existing sensor networks and applications can be reused as well as allowing access from the outside to be controlled.

2.4. Methodology

The methodology applied throughout this work, as with much ubiquitous computing research, is experimental. This fits well with the desired focus on applied work (See above, section 2.2).

The employed research process has been inspired by the rational unified process, as described by Kruchten [Kruchten 2003], but adapted for research. It can be broken down into five phases: problem, requirements, proposal, experimentation, and evaluation. The first phase reviews the work in the field and defines the area of study and the problem. The second phase captures the requirements of the defined problem. The third phase designs a proposal that addresses the problem by fulfilling the requirements. The fourth phase implements the proposal and tests it experimentally. The fifth and last phase evaluates the solution proposal and the experiments performed. Figure 4 illustrate the research process.

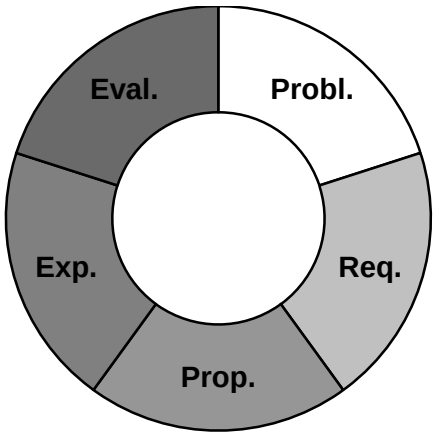


Figure 4. Research process

The process has been performed iteratively. This yields a gradual research process with room for feedback.

2.5. Infrastructure approach

There are different ways in which privacy protection and development support can be provided.

The most straightforward approach is to provide separate tools for the tasks. For example, proxies can be used to improve users' privacy by filtering their requests, blocking those that would result in a violation of the users' privacy

[JRC P3P Resource Centre 2005A] whilst development support can be provided with libraries providing APIs to desirable functionalities [Java Community Process 2003]. This separation is, however, not desirable. Firstly, it would be inefficient to manage and maintain different tools. Secondly, the tools may end up working against each other, due to the unavoidable conflicts that exist between context-aware functionality and privacy. Hence, a common instrument, addressing both issues, is preferred.

The approaches taken in early related work has been to provide developers with the basic building blocks necessary to develop complete context-aware systems. Schilit, for example, presents a context-aware computing architecture [Schilit 1995], whilst Dey describes a framework and toolkit [Dey 2000B]. Hong and Landay, however, argue that the use of an infrastructure approach is more beneficial [Hong, Landay 2001]. An infrastructure provides further abstraction for application developers allowing them to focus on the problem at hand. Examples of work that has opted for an infrastructure approach includes the MobiComp infrastructure [Ryan 2005], the solar system [Chen, Kotz 2002], and EQUIP [Greenhalgh 2002].

The use of an infrastructure has three key advantages according to Hong and Landay [Hong, Landay 2001]. Firstly, by using services in an infrastructure context-aware applications can be developed independently of the platform, assuming the interaction with the services are standardised. Secondly, an infrastructure provides a middleware layer that can separate the capture, distribution, and use of context information allowing components to be developed and maintained independently. Finally, an infrastructure enables resources to be shared between applications.

With respect to privacy protection it is also believed that the use of an infrastructure approach can be beneficial. By developing the infrastructure to handle the privacy protection, this responsibility is lifted from the applications and their developers. It is deemed that this can simplify the development of applications. Furthermore, by incorporating the privacy protection into the infrastructure, better possibilities ought to exist to provide a uniform protection mechanism.

2.6. Related Work

Given the width of the combined field in which this work has been undertaken, this section will only present a selection of the related work. The focus will be on aspects considered particularly important when developing a privacy-enhancing infrastructure.

2.6.1. Context and Context-awareness

Different definitions of context and context-awareness have been used in related work, each emphasising aspects found to be important.

The early work in the field by Schilit and others describes context to capture changes to things of interest, specifically they mention three aspects they consider important: "where you are, who you are with, and what resources are nearby" [Schilit, Adams et al. 1994]. Hence, change characterises contextual information. Schilit et. al. also continue to present a number of examples of what they consider to be contextual information, including location information, lighting conditions, noise level, connectivity, communication costs and bandwidth, and information about the social situation.

The importance of changes is further emphasised by Schilit's and Theimer's definition of context-aware computing. They define context-aware computing as "the ability of a mobile user's applications to discover and react to changes in the environment they are situated in" [Schilit, Theimer 1994 p.3]. Thus, the definition stresses the importance for context-aware systems to be able to capture and utilise contextual information.

A later definition of context presented by Dey and Abowd stresses the need for information to describe a situation relevant to a user-application interaction (See above, section 1.4). Hence, Dey and Abowd focus on the relevance of information rather than on whether the information simply changes or not. They also present four primary context types: location, identity, time, and activity [Dey, Abowd 2000A]. Contexts that do not fit into those types are considered to be secondary and are indexed by the primary types in information space.

Dey and Abowd also provide a definition for context-aware computing (See above, section 1.4). The definition is similar to that presented by Schilit and

Theimer in that it stresses the use of context information. However, what makes Dey's and Abowd's definition distinct is that they continue to emphasise the importance of relevance instead of changes.

These definitions illustrate there that there are differences in how context and context-awareness is defined, changes vs. relevance. However at the same time the definitions can also be seen to overlap under certain conditions, for example, when changes are relevant.

2.6.2. Application domain

A fundamental idea of ubiquitous computing is that computational devices should be ever-present in our environment. It is therefore not surprising that the domain covered by related work is large.

The early work in the area was carried out in office environments [Weiser, Gold et al.1999] [Want, Hopper 1992B]. Since then there have been work carried out that focus on peoples' homes [Kidd, Orr et al. 1999] [Intille, Larson et al. 2005] and places of leisure [McCarthy 1998] [Kindberg, Barton et al. 2002]. Some have also gone outdoors to cover such domains as tourist sites [Cheverst, Davies et al. 2000] [Mobil Turism 2005] and archaeological work [Ryan, Pascoe et al. 1997]. Furthermore, a large proportion of systems are mobile and work without any fixed infrastructure [Ryan 2005] [Osbakk, Rydgren 2005].

In each of the environments different types of applications and services have been deployed. For example, the active badge project provides a call forwarding service at the office [Want, Hopper 1992B], MusicFX provides a personalised music service at the gym [McCarthy 1998], and the GUIDE project provides information to visitors information of interest [Cheverst, Davies et al. 2000]. These are just some examples of different services that can be and are provided by ubiquitous computing systems.

It is therefore not possible to isolate ubiquitous computing to any single environment. Furthermore, there is no single application or service that characterises ubiquitous computing and can be used as a standard template.

2.6.3. Devices used

A wide range of devices are being used in ubiquitous computing systems including both commercially available consumer devices and specially designed experimental platforms. Of interest to this work are four types of devices: TabletPCs, PDAs, Smartphones, and embedded devices.

The TabletPCs are the most powerful of the four types of devices. They are perhaps best described as slimmed down portable personal computers. Some features distinguishing the TabletPCs from ordinary laptops are lower weight, smaller size, and the addition of a pen-based mechanism for data entry. Both the form factor and the mechanisms for interaction of the TabletPCs are remarkably similar to the Pads described by Weiser [Weiser 1993]. The capabilities of a TabletPC are comparable to that of a low-end laptop. Thus, they generally provide abundant processing power, memory, and disk storage for ubiquitous computing applications. An example of related work using early TabletPCs is the GUIDE project [Cheverst, Davies et al. 2000].

The PDAs are perhaps the most commonly used device type, out of the four. They provide a general purpose computing platform in a handheld package. Similar to the TabletPCs, PDAs generally provide a pen-based mechanism for data entry. In contrast to TabletPCs, a PDA's processing power and memory is much more limited. Furthermore most PDAs do not feature disk storage but have to make do with limited amounts of flash memory. However whilst the PDAs are not on a par with the TabletPCs in terms of performance, their small size and low weight make them much more mobile. In terms of the shape and size these devices are similar to the tabs described by Weiser [Weiser 1993]. Their high cost, though, so far prohibit them from being scattered around a user's environment as envisaged. Related work using PDAs includes FieldNote [Ryan, Pascoe et al. 1999], Cooltown [Kindberg, Barton et al. 2002].

The Smartphone is a device type that has gradually become more interesting with respect to ubiquitous computing as the possibility to run third-party applications has improved. A smartphone is essentially a mobile phone to which PDA functionality has been added. They are however often more limited than PDAs in terms of performance and capabilities. For example processing power

is seldom prioritiesed in a smartphone and it is not uncommon that they lack pen-based input. Smartphones however have the advantage of featuring a mobile-network connection. This generally provides wireless connectivity with a better coverage. It also allows consumers to make voice calls and run applications on a single device. Smartphones, just like PDAs, can be seen as an expensive tab. Related work using smartphones include MobiTip [Rudström, Svensson et al. 2004], Mobil Guide [Mobil Turism 2005], Bluereminder [Osbakk, Rydgren 2005].

The last type, embedded devices, is frequently used when building sensor networks and ubiquitous computing artefacts. An embedded device is a computational platform that is committed to a specific task and closely tied to a surrounding system. Generally, and particularly in ubiquitous computing, embedded devices have a small form factor. The actual specifications vary greatly from device to device but common characteristics include the lack of a user interface of their own and severe constraints on resources, even more so than with smartphones or PDAs. Some examples of embedded devices used in related work are SmartIts [Gellersen, Schmidt et al. 2002], TINIs [Russo, Sukojo 2004], and Mica motes [Hill, Culler 2002].

The capabilities of all of these four types of devices continuously improve. However, it is still possible to get an idea of the differences that exist by inspecting a sample of the available devices. The table in Figure 5 shows the capabilities of one device from each of the four types.

Device	CPU	Memory/ Storage	Connectivity	Size (mm)/ Weight (Kg)
A) Tecra M4	Centrino M740 1.73 Ghz	512 Mb / 60 Gb	IR, Wifi, Bluetooth, Ethernet	328x290x38‡ / 2.8
B) Ipaq 4150	ARM PXA255 400 Mhz	64 Mb† / (Exp. card)	IR, Wifi, Bluetooth	114x70x14‡ / 0.132
C) Sony Eric. P910i	ARM 9	64Mb† / (Exp. card)	GSM (GPRS)	115x57x26 / 0.155
D) DSTINI m400	DS80C400	1Mb / 1Mb	Ethernet	67x48x5‡ / ?

† Shared memory and storage ‡ Approximate measures
A) [Toshiba 2005] B) [Hewlett-Packard Company 2003]
C) [Sony Ericsson 2004] D) [Dallas Semiconductor, Maxim 2005A]

Figure 5. Overview of device capabilities

2.6.4. Privacy protection

With the development of electronic commerce, online privacy has become an acute issue. To guide service providers five principles central to fair information practises have been composed: notice, choice, access, integrity, and enforcement [Landesberg, Levin et al. 1998]. Firstly, subjects should be made aware when their information is collected how it will be used. Secondly, they should be given the choice of whether to participate or not. Thirdly, subjects should be able to access and correct information held about them. Fourthly, captured data should be accurate and kept securely. Finally, it is necessary for the principles to be enforced. These principles capture the basic requirements for privacy protection.

In ubiquitous computing privacy issues are even more prominent, especially when combined with context-awareness. Research into the adoption of privacy protection for this field, however, has been and still is limited. Nevertheless, some projects have been undertaken on this topic. They investigate techniques that may allow ubiquitous computing systems to conform with the principles of fair information practises as well as other methods for the protection of privacy.

From the principles of fair information practices six design principles for preserving privacy in ubiquitous computing system have been derived [Langheinrich 2001]. These include notice, choice and consent, anonymity and

pseudonymity, proximity and locality, adequate security, access and recourse. Based on these design principles the privacy awareness system [Langheinrich 2002] has been developed. The central idea in the project is that privacy beacons are used to announce where and when information may be collected by a service. These announcements are picked up by a privacy assistant carried by the users and forwarded to a remote but personal privacy proxy. The privacy proxy analyses the situation by inspecting the service's privacy policy and compares it with the user's preferences. A choice can then be made whether to utilise the service or not. The privacy proxy also keeps track of services used and the information collected. This allows the proxy to send updates to services when relevant information changes. Furthermore the privacy awareness system is also stated to provide secure communication and access controls. Hence, the system covers all the fair information practice principles except enforcement.

The enforcement of privacy policies is a problem that due to its difficulty perhaps never will be completely addressed. However, the use of social mechanisms, such as reputation and trustworthiness, allow the situation to be improved upon. Goecks and Mynatt describe a personalized reputation system for the protection of privacy in ubiquitous computing environments [Goecks, Mynatt 2002]. The idea is that every user has a reputation built from the trust placed in them by other users. The trust network is weighted such that the opinion of trusted users weighs more than those that are not trusted or unknown. Once calculated the reputation can then be used as a means of ascertaining a user's trustworthiness and thus whether it is desirable to share information with them or not. Since the feedback left by a user also affects the larger network, malicious acts can be penalised in this type of system by reducing the trust placed in the rogue user.

An alternative approach to that of adopting the principles of fair information practices has been taken in the solar system [Minami, Kotz 2002], an infrastructure supporting the capture, processing, and distribution of contextual information [Chen, Kotz 2002]. In this system the focus is on controlling the release of contextual information using access controls rather than on its use once released. In solar, contextual changes are represented as events and

applications subscribe to event streams to retrieve information. To control access, each event in solar is tagged with an access control list (ACL). The ACL specify who can access the event as a list of named principals or roles and is created together with the event. As the events flow through the system, their associated ACLs are modified appropriately to reflect any transformation the information undergoes. The access can then be enforced by restricting the delivery of events to applications that execute on behalf of a principal listed, either directly or through role memberships, in the event's ACL. Thus, assuming that the ACL is true to a subject's privacy preferences then their privacy is protected.

2.6.5. Security aspects

Providing a subject with the ability to control their flow of personal information is of little use unless the infrastructure and its communication is secure since without adequate security, sensitive information may fall into the wrong hands anyway. Security is therefore a necessary condition when addressing privacy, though it should be noted that it is not sufficient alone.

The book S*ecurity for Ubiquitous Computing* [Stajano 2002] provides a comprehensive discussion of the security issues that exist in ubiquitous computing. In the book Stajano examines five aspects deemed important for security. Firstly there is confidentiality. A secure system needs to keep information secret to all but the intended recipient(s). Stajano highlights the increased vulnerability of wireless networks. Given the absence of wires, the opportunity to listen in on communication exists for anyone within signal range. Secondly there is integrity. Information must be protected against unauthorised modifications for a system to be secure. Stajano makes the point that integrity applies not only to information being transported, but also to information held by hosts. Thirdly there is availability. In a secure system rogue users must not be able adversely affect the availability of the system and deny legitimate requests from being processed. Fourthly there is authentication. It must be possible to verify the identity of users. Stajano emphasises the importance of authentication for security and states that confidentiality, integrity, and availability can be violated without it. Finally there is anonymity. Stajano draws

attention to the fact that it is not always the content of messages that yields the most useful information but its existence.

These five security aspects need to be taken into account when developing any ubiquitous computing system and especially if the system is to protect the privacy of its users.

2.7. Summary

In this chapter the basis for this thesis on privacy in context-aware ubiquitous computing has been introduced.

The motivation behind this work concerns the issues that exist in developing publicly acceptable context-aware ubiquitous computing systems. In particular the work has been motivated by the need for improvements in privacy protection and development support.

Given the wide scope of possible research within the field of study, the focus of the project has been constrained to applied experimental research on the software side of ubiquitous computing. Furthermore, to avoid a constantly moving target the platform and the technologies used have been kept constant whenever possible.

The aim of the research is to contribute to the ongoing development in the combined field of privacy, context-awareness, and ubiquitous computing. In particular the intention is to contribute to the improvement of users overall privacy, the evaluation of different access control mechanisms, the provision of development support, and the protection of third-party infrastructures.

The research process used has been inspired by the rational unified process and consists of the five phases: problem, requirements, proposal, experimentation, and evaluation. Together the phases provide the necessary structure to perform the research. In this work the research process has been performed iteratively.

In this work an infrastructure approach for providing privacy protection and development support has chosen to be investigated. The choice is motivated by the benefits Hong and Landay argue to exist including platform independence, separation of components, resource sharing. Another motivation is that it is

believed to be beneficial to place the responsibility for privacy protection on an infrastructure rather than individual applications.

Also described in this chapter is related work in a number of areas including context-awareness, application domain, devices used, privacy protection, and security. What is evident from the work described is ubiquitous computing is a truly diverse field, without any established standards. It is therefore especially important to not assume implicit knowledge.

This leads us to the next chapter, which will present the definitions of context and privacy employed in this work. The chapter will also describe the conceptual models of context and privacy derived from the definitions.

CHAPTER 3

CONCEPTUAL MODELS

Neither context nor privacy are concepts that are without ambiguity. This is clearly undesirable. To develop a privacy-enhancing infrastructure for context-awareness a firm understanding and formal definitions of both these concepts is essential.

This chapter will present the conceptual models that form the base of this work. Two models have been put together, one for context and one for privacy. The context model emphasises the existence of relationships between entities. This creates networks of contextual information. The privacy model defines privacy in terms of control over a subject's flow of information. It also states that the ideal level of privacy in ubiquitous systems is equal to that enjoyed offline.

3.1. Context Model

Although most research done in the context-aware field takes a similar view of what constitutes context, the exact definition used varies from project to project. In the selection of work presented in the background chapter everything from location to musical preferences has been referred to as context. Whilst the variations in the exact definition of context seldom cause any practical problems it is still important to be clear about what specific definition is in use.

This section is therefore dedicated to presenting the context model employed in this work. It will start by presenting the definition of context used in this work. After this the derived model of context will be described. Finally the section will show how this model can be represented both graphically and textually.

3.1.1. Definition

The definition of context employed in this work is broader than either of the definitions described previously (See above, section 2.6.1).

"*Context is information related to an entity, where the information may be an entity itself*" [Osbakk, Ryan 2003]

The definition stresses the existence of a relation between entities and data values. Compared to the definitions previously described, it is closer to the linguistic definition of context which defines context as "associated surroundings" [Larousse 1994]. It is therefore hoped that the definition will better fit the concept of context that is held by the general public.

By requiring the existence of a relation the definition also rationalises why one piece of information can be considered to be context under some circumstances but not under other. For example take the string "ISBN 0-7522-2470-0". This string represented on its own would not be classified as context information. But if I state that the string provides a reference to the book "The Dilbert Principle" by Scott Adams, the very same string will now be considered to be a piece of context as its relationship to an entity has been defined. This context relation is illustrated in Figure 6.

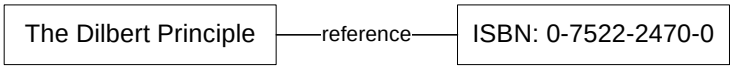


Figure 6. An entity to context relation

It is also important to note that the definition does not place any limitation on what an entity must be nor on the type of information that constitutes context. Thus given the existence of a relation, anything can be considered to have and/or be context. This includes physical entities such as people, places and things [Kindberg, Barton et al. 2002] as well as virtual entities such as events and concepts. For example, assume that a meeting is held where Alice and Bob are present. Perhaps they are exchanging secret keys. This meeting will thus be a piece of context with a relation to both Alice and Bob, e.g. current activity. But the meeting itself can also be regarded as an entity on its own. It is a virtual, and in this case a temporal, entity. Thus the meeting can also have context

information associated with it. Obvious examples of such context information would be the attendees, time, purpose, location etc. Figure 7 illustrates a set of entity to context relations.

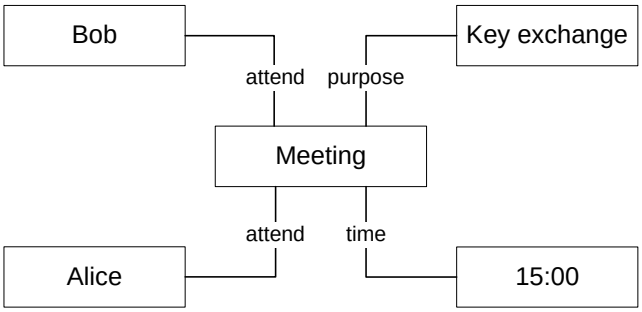


Figure 7. A set of entity to context relations

Finally the width of the definition ensures that narrower application specific definitions can coexist within a system. For instance the definition by Dey and Abowd [Dey, Abowd 2000A] needs the usage of a piece of information to be clear whereas knowing the type of information is crucial with the classification employed by Schilit et al [Schilit, Adams et al. 1994]. Thus situations exist where a piece of context will be considered context in one system but not the other. By using the definition presented here and then a narrower application specific definition only when necessary, the information may coexist within an infrastructure (the existence of a relation is intrinsically assumed given that unreferenced information is seldom valuable and thus rarely used). This feature will later be shown to be important for the integration of the developed infrastructure with others.

3.1.2. Network of relationships

The context model used in this work is directly derived from the employed definition of context. Consequently it takes the shape of a network, where the network is created from the relations between entities and pieces of context (or other entities). The previous figures have shown very limited context networks, starting with simplest containing only one entity and one piece of context.

The world we live in however is much more complex, hence we will get an almost infinitely complex network of relationships when using this model. The network will represent the context of any entity within it. For example assume

that Alice has read the book “The Dilbert Principle” by Scott Adams [Adams 1997] and so has her friend Bob. Thus a relationship exists between them and the book as well as between themselves. Figure 8 is a graphical representation of the context model derived from the information given so far.

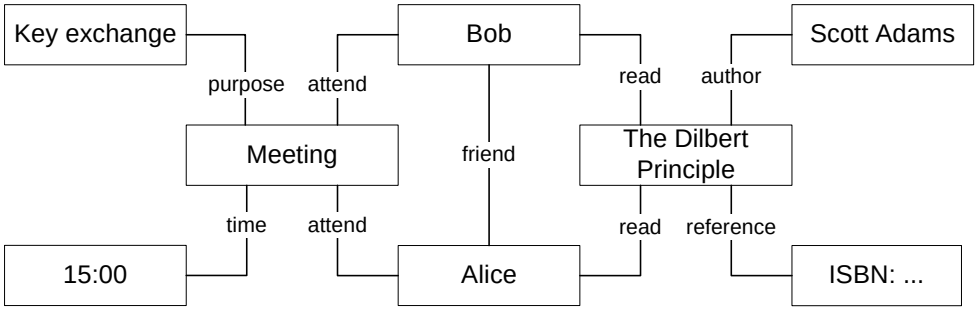


Figure 8. A simple context network

What becomes clear from the above example is that the relationships work both ways and those entities that initially may seem unrelated such as Bob and Scott Adams form part of each other’s context.

The context of any one entity in the model is found by resolving the associated network of relationships and may include recursive relationships. In a real world situation this will create a graph of arbitrary size, where the size will depend on the number and formation of the relationships. For practical reasons it will thus be expected that the length of the relationship chain will be fixed to a manageable value during modelling. Although this can reduce the model to a manageable size it introduces the problem of selecting the scope. The decision on what scope to use will largely be subjective, where important factors include the intended use, requirements, and the available resources. Thus the decision needs to be made on a per model basis where a trial and error approach may be employed. In an application scenario on the other hand the graph size would be drastically reduced as privacy protection mechanisms should, and is assumed, to limit the length of the chain. Thus it may not be necessary to further reduce the context model.

3.1.3. Representation

So far the context model has only been represented graphically. Although useful when rendering an overview of the context network, it is not adequate when the

model is going to be processed. The model ideally needs to be represented in a format that allows it to be read and interpreted by both humans and machines. So how do we do this? The approach taken in this work is to represent the context-network using the Resource Description Framework (RDF) [World Wide Web Consortium 2004A]. An approach also taken by others [Korpipää, Mäntyjärvi et al. 2003] [Korpipää, Mäntyjärvi 2003].

RDF is a framework developed to allow information about web resources to be represented [World Wide Web Consortium 2004B]. It is based on the idea that resources are associated with properties and data values which can be described using simple statements. Such statements consist of a subject, predicate, and object. The subject identifies the resources about which the statement is made and consists of a Uniform Resource Identifier (URI). The predicate refer to the particular property being described. Finally, the object is the value of the property, which can be a literal or a resource itself. These statements can then be represented using an RDF graph where the predicates form arcs between subject and object nodes, as illustrated in Figure 9.

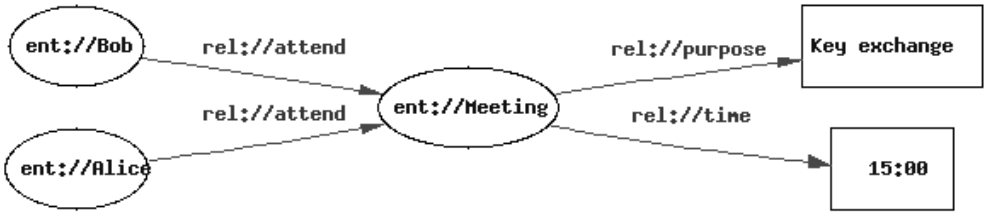


Figure 9. RDF Graph representing a simple context model

Although RDF is focused on representing web related metadata it is not limited to this use. RDF can represent information from other areas too. The only requirement is that the resources are identified using URIs. The translation between the employed context model and the RDF representation is straight forward: entities translate on to resources, relations to properties, and context to data values. As can be seen, there is a good match between the context and RDF models. The context model has indeed been influenced by RDF. However, there are subtle differences. The RDF model always makes a distinction between data values and resources. The graph representation makes this distinction clear, showing resources using ovals and data values with rectangles. Although accurate given the current data set, it should be remembered that the

length of the relationship chain may be limited. Thus the data values may in reality be entities themselves, which is why no distinction was made in the previous graphical representation of the context model. Note the namespaces used here are abbreviated to improve the legibility.

As previously stated the reason for using RDF is the need to represent the context model non-graphically. The most intuitive way of doing this using RDF is as a collection of subject – predicate – object triples. This allows a condensed listing of the current information in the model to be provided, as illustrated in Figure 10.

Subject:	Predicate:	Object:
ent://Bob	rel://attend	ent://Meeting
ent://Alice	rel://attend	ent://Meeting
ent://Meeting	rel://purpose	“Key exchange”
ent://Meeting	rel://time	“15:00”

Figure 10. RDF triple representation of a simple context model

Alternatively the context model can be represented using RDF serialised as XML (RDF/XML) [World Wide Web Consortium 2004C], see Figure 11. Even though the structure of RDF/XML is more verbose than RDF triples, this is the preferred format. A key benefit of RDF/XML is that the format is well defined. By using services such as the W3C RDF Validation Service [World Wide Web Consortium 2005A] the validity of such a document can thus be established. RDF/XML is also the commonly used representation of RDF. Furthermore both RDF graphs and RDF triples can easily be obtained from it, e.g. using the former service.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
    xmlns:schema="rel://">

    <rdf:Description rdf:about="ent://Bob">
        <schema:attend rdf:resource="ent://Meeting"/>
    </rdf:Description>

    <rdf:Description rdf:about="ent://Alice">
        <schema:attend rdf:resource="ent://Meeting"/>
    </rdf:Description>

    <rdf:Description rdf:about="ent://Meeting">
        <schema:purpose>Key exchange</schema:purpose>
        <schema:time>15:00</schema:time>
    </rdf:Description>

</rdf:RDF>
```

Figure 11. XML Serialised RDF representation of a simple context model

Finally it should be mentioned that although RDF does not directly provide any methods for describing the relationships between the represented resources beyond their existence, the related RDF vocabulary description language [World Wide Web Consortium 2004D] does this. This language allows the creation of RDF Schemas that define the vocabulary used in RDF documents. By using these schemas the validity of RDF documents and hence the represented model can be established.

3.2. Privacy Model

The need for privacy in context-aware and ubiquitous computing environments has been stated to be a key motivation for this work. The need for privacy is not exclusive to these environments, nor to today's society. The work by Westin [Westin 1970] that has been presented in the background chapter argues that this desire can be traced back as far as our animal origin. However the introduction of this new technology enables the invasion of privacy on a much larger scale and using fewer resources than before. Although the issues concerning privacy were encountered already with the early work in ubiquitous computing the work in this area has been limited. Thus there is no well established model of privacy, leaving most ideas subject to discussion.

This section will therefore introduce the conceptual model of privacy used as a basis for the work presented in this thesis. The aim is to clarify the position

taken with respect to privacy. It will also later allow requirements to be captured. The chapter will start by presenting the adopted definition of privacy. It will then continue to discuss what is thought of as being the ideal level of privacy followed by the subject's scope of control. Furthermore, the section will present the process of disclosure in the model as well as how the concept is visualised. Finally, an overview will be given of how this privacy model relates to the employed model of context.

3.2.1. Definition

The linguistic definition found in a dictionary [Larousse 1994] refers to privacy as "seclusion", "freedom from intrusion by the public", "avoidance of notice", etc. This gives the impression that it is necessary to protect a subject from all forms of external interaction to achieve privacy. As such, the ability to participate in the society would be limited. Such behaviour may in turn be interpreted as abnormal and be taken as an indication of something being wrong. Indeed the dictionary [Larousse 1994] also refers to privacy as "secrecy" and "concealment", which is easily associated with negative behaviour.

The work presented in this thesis, however, views privacy from the perspective of information flow. This is natural given the nature of the area in which privacy is considered. One of the key differences between traditional and context-aware applications is in fact that the latter takes advantage of the information available about its surroundings. Thus throughout this work the following definition of privacy has been adopted:

> *"Privacy is the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others"* [Westin 1970 p.7]

The definition emphasises the subject's right to control the flow of information about them. Privacy does not therefore equal isolation nor does it imply that there is such a desire. The issue is instead one of control. A subject may choose to make the flow of information about them fully public or they may indeed choose to isolate themselves. What is important is that the subjects themselves

can control the extent of their information flow. As such it is with the subjects that the ownership of information rests.

Thus, the availability of privacy does not limit the participation in society by default, though a subject's choice may. The majority however are expected to strike a balance between the two extremes of complete openness and complete isolation. There are many factors that may influence this balance of privacy, among those mentioned in *Privacy and Freedom* by Westin [Westin 1970] are for example the political system in use, historical and cultural traditions, life situation, personal preferences, etc. From this list it can be seen that some of the factors can be affected by the subjects and not others. This work will therefore make a distinction between the range of privacy society allows and the particular trade-off a subject makes. The former will be referred to as the *level of privacy* and describes how much control a subject has in determining their balance of privacy. The latter will be referred to as the *effective balance of privacy* as it represents the subject's choice of privacy within the range specified by society. It is therefore the effective balance of privacy that can cause a subject's claim for privacy to be interpreted negatively by others.

3.2.2. Ideal level of privacy

What level of privacy is then required? The position taken in this work is that the introduction of new technology should not adversely affect people's privacy. It does not matter if the effect on privacy is limited. Over time individual and even seemingly small sacrifices will add up to eventually give rise to an undesirable level of intrusion. We should also not automatically assume that the benefits provided by context-aware environments will justify any reduction in privacy. The current level of privacy is therefore thought to be the ideal and should therefore remain static with the introduction of new technology.

The reference point, against which the level of privacy is fixed, has been chosen to be that of an offline environment. This choice has been made because of two reasons. Firstly, the nature of an individual's interactions with the physical world is considered to be mainly offline, i.e. they do not occur over a computer network. Most interactions take place directly between an individual and non-networked objects or other people. Offline is therefore thought to be the natural

state against which privacy needs to be compared. Secondly, the level of privacy people enjoy while being offline is found acceptable by most. This is not surprising since the established balance of privacy in our society is in part determined by tradition and culture and thus has gradually evolved over many years. Also, in recent years there have not been any significant changes to the western world's politically system, a key determinant of the balance. This has also contributed by allowing people to grow accustomed to the privacy that is available. Thus this work makes the assumption that:

The level of privacy in context-aware ubiquitous environments should ideally be equal to the privacy enjoyed whilst being offline.

But what level of privacy does an offline environment then provide? Consider the scenario previously described where Bob and Alice meet to exchange cryptographic keys. In this situation the information exchange is voluntary and under their control. Either one could decide not to share their information if they wish to do so. However they have less control over the fact that the meeting is taking place. If a third-party sees Alice and Bob entering the meeting room the occurrence of a meeting can be deduced. Alice and Bob do therefore not enjoy complete privacy, they do not have full control over their information flow. The offline world is therefore considered to constantly leak private information. Furthermore, there are no absolute guarantees of how the information exchanged during the meeting will be used. Even though there is a relation of trust between Alice and Bob that may stipulate that they will not reveal confidential meeting information, either of them can at a later stage break this trust. Therefore, once information has been revealed in the offline world, no control is held over it by the person disclosing it. This is also the case with leaked information, as once acquired the third-party may continue to spread the information. The privacy enjoyed in an offline environment is therefore not perfect. Indeed the actual level of control a subject has in an offline environment is fairly limited and can be summarised as being restricted to controlling their own disclosures, their presence, actions in public spaces, and receptiveness. These areas will be further discussed later.

What is important to note is that even in simple situations, free from invasive technology, information can leak and thus potentially be misused. Thus the intrusion caused by information leakage in an offline environment is part of people's everyday life and there is, in practise, little that can be done to avoid this. The level of privacy that is needed in an ubiquitous system does therefore not necessarily need to be perfect, rather the requirement in the adopted model is for an adequate level of privacy to be provided. The privacy enjoyed and accepted in our everyday offline environments is after all imperfect. Indeed Westin argues that "the individual's desire for privacy is never absolute, since participation in society is an equally powerful desire" [Westin 1970 p.7]. This does not mean though that privacy cannot be enjoyed whilst participating in society but that it is difficult for an individual to *fully* control the flow of information about them in such situations. Thus a certain degree of privacy invasion will be deemed as acceptable.

3.2.3. Scope of control

At the start of this chapter privacy was defined as a subject's right to control the flow of information about them, making the scope of control the primary factor that determines privacy. This section will now further describe the extent of the control a subject has over their flow of information.

Two fundamental assumptions have been made about the scope of control, both briefly introduced in the previous section:

- Information constantly leaks from a subject's presence and actions in public spaces
- Control over information can only be exerted up to the point of disclosure.

The first assumption specifies that there is a constant leakage of private information. Even if a subject does not choose to disclose any information, their presences and actions in public spaces may allow others to deduce certain pieces of information. The second assumption specifies that the control over the flow of information can only be exerted up to the point of disclosure. Once information has left a subject's personal space, be it by a deliberate disclosure

or an uncontrolled leakage, the subject cannot control how it will be used. This relaxation simplifies the privacy model to one that is practically achievable in an imperfect world.

Because of the limitations incurred by these assumptions the level of control a subject has over their information flow is practically limited to the four areas:

- Controlling their own disclosures
- Controlling their presence or recognition in public spaces
- Controlling their actions, and link therewith, in public spaces
- Controlling their receptiveness

Firstly, controlling their own disclosures is, in terms of this work, the foremost way in which a subject can enforce their privacy. By directly controlling what information is disclosed and to whom any disclosures are made, the subject can guard their private information. Such control is broken down into two approaches:

1. A subject may choose to keep the existence of certain pieces of information secret.
2. A subject can also just reject requests for private pieces of information.

In (1), the subject may avoid having to deal with requests for the information kept secret. For example if Bob does not acknowledge having taken part (and having knowledge of what was said) in the meeting with Alice, a third-party is not expected to ask him what happened during the meeting. This however is dependent on two conditions (A) that the existence of the information is indeed a secret and (B) that the subject does not unveil the secret when queried arbitrarily. If either of these conditions are false then this approach does not work. Such breakdown can occur if for example someone sees Bob entering the meeting, or if someone knows Bob frequently attends such meetings with Alice. Also in the cases where the subject denies knowledge of information and the third-party knows for a fact that this is untrue, consequences may follow either due to the dishonesty or the withholding of information.

In (2), following the example given so far Bob would then make no effort to hide his attendance at the meeting; instead he would tell a third-party the meeting was private. This approach is not limited by the two conditions of the previous paragraph as there is no denial to possessing the information. This however implies that the requester always knows that the subject is withholding information, and the respective consequences may follow.

Secondly, a subject can also control their presence or recognition in public spaces to minimise the leakage of private information. In this way a subject can proactively manage their privacy, even in situations where there is no direct control over the information flow. This area can also be broken down into two approaches:

1. A subject can avoid being present in public spaces where the information leakage is deemed to be potentially damaging.
2. A subject can instead of controlling their presence, hide it.

In (1), for example assume Bob is interested in taking up employment with a rival to his current employer and that they hold a public event. By not attending this event Bob can ensure his intention does not leak, thus protecting his privacy. This approach of avoiding information leakage, although assured, is not always desirable as it negatively affects a subject's freedom.

In (2), by remaining anonymous the information leakage will not be meaningful, thus protecting the subject's privacy. Indeed in many situations a subject will be anonymous as they will only be seen as being part of a larger crowd. Thus if the attendance to the event is high enough then Bob may successfully take part while still remaining anonymous to the other attendees. Although this approach allows greater freedom for the subject, success is dependant on the condition that the subject needs to remain anonymous with respect to the particular public space. This in turn requires the subject to be inconspicuous and limits their possibility to interact. If this condition is not met and their identity is discovered or deduced, either while present or later, the information leaked will become meaningful. Thus consequences may follow.

Thirdly, by controlling the actions taken, and their link therewith, a subject can control the leakage of information in public spaces even when not hiding their presence. This will allow a subject to interact with other people yet have some control over the information leaked and thus their privacy. Similar to before, this area is divided into two approaches:

1. A subject can actively control the action they take in a public space to ensure the information that leaks will not be too invasive.
2. A subject can also conceal the link between them and their actions, e.g. by using a pseudonym.

In (1), for example, if Bob attends a rival company's event then by controlling his action there, e.g. not openly meeting their recruitment personnel, his intention to changes sides may remain private. Once again this will of course limit the freedom of the subject.

In (2), this limits the meaning of the information to the particular pseudonym and, if desired, public space. If Bob is approached by another attendee he can use a pseudonym if he needs to identify himself. This will allow him to be known during the interaction (and between separate occurrences of interaction) yet isolating the leakage of information. This approach therefore allows greater freedom for the subject to interact. However the success of this approach depends on the subject remaining unlinked to their pseudonym and thus their action. If this condition is not held the information leak can be linked with the subject and consequences may follow.

Fourthly, by controlling their receptiveness to information a subject can introduce uncertainty in the information flow. This will ensure plausible deniability if the subject later is confronted with the information due to a leakage. It can also reduce the risk of incorrectly disclosing the information themselves in the future. Two approaches have been identified in this area:

1. A subject can treat unsolicited information as noise.
2. A subject can actively select which information to acknowledge.

In (1), by treating unsolicited information as background noise it can be ignored. For example, when Bob is at the rival company's event he can focus entirely on the information he is interested in, ignoring what happens around him. If he is observed and an attempt is made to communicate with him it will go unnoticed. This hopefully causes the observer to become uncertain of whether the observation was correct or not. Bob can then insist that a mistake must have been made if later confronted with the information. This approach requires the subject to be in full control of their receptiveness to information, which may not always be possible. However, when successful, the subject has no knowledge of the unsolicited information.

In (2), should Bob fail to successfully ignore the attempt to communicate he can opt not to acknowledge it. Again, this will enable him to insist a mistake must have been made. This approach requires the subject to suppress the natural reflex to respond when an attempt is made to communicate with them. It should be noted that persistently ignoring other's attempts to communication can be interpreted as anti-social behaviour and may incur consequences from those being ignored.

These four areas represent the actual control a subject holds. Because of the limitations imposed by the assumptions made, a subject does not have full control over the information communicated about them. This constrains the level of privacy that is achievable within the model and may indeed limit the possibilities a subject has of achieving the effective balance of privacy they desire. But even so it should be possible to achieve what has been defined as the ideal level of privacy simply because these assumptions are deduced from the offline environment in which we live. Furthermore, it will now be shown that all of the four basic states of privacy (Solitude, Intimacy, Anonymity, and Reserve) described by Westin [Westin 1970] can be attained with the available scope of control. Figure 12 outlines the mapping.

	Solitude	Intimacy	Anonymity	Reserve
Own disclosures				√
Presence or recognition	√	√	√	
Actions or links			√	
Receptiveness				√

Figure 12. Mapping to Westin's four basic states of privacy.

The first state, solitude, is characterised by the subject not being part of any group or under any observation. Thus a subject can attain this state by limiting their presence to spaces that are known to be secluded and safe. The second state, intimacy, is characterised by the subject being part of a small group separated from the rest of society. This state can therefore also be attained by the subject by controlling their presence, but this time to a space controlled by the group. The third state, anonymity, is characterised by the subject remaining unidentified and unknown. This state can thus be attained by the subject either by hiding their identity or by presenting an un-linkable identity, i.e. a pseudonym. The fourth state, reserve, is characterised by the use of "physiological barriers" [Westin 1970 p.32] to retain privacy, where such barriers may involve the limitation of communication about themselves. This state is therefore attained by the subject controlling their own disclosures and by controlling their receptiveness. Hence all of the four states are indeed attainable.

3.2.4. Process of disclosure

Because of the imposed limitation of not being able to control personal information once released, the process of disclosing information plays an important role. It is only within this process that the subject can actively control their effective balance of privacy, using the three areas of control previously discussed. In this conceptual privacy model the process of disclosure is therefore thought to be like a business transaction where the disclosure of information follows some form of agreement that governs how the information will be used.

3.2.4.1. Participants

There are two types of participants that can take part in a transaction. First, there is the subject, the primary participant, whose personal information the transaction is about. Their aim it to ensure that the release of information only occurs on terms that do not violate their privacy preferences. Then there are the potential recipients of information, the secondary participants. In many cases it will be they who initialise the process of disclosure with a request. Their aim is of course to persuade the subject to release information for the purpose they require at as favourable terms as possible. For a transaction to be able to occur there must be at least one primary and one secondary participant. By definition only one primary participant can take part since one subject alone should control their release of information. However in certain situations there may be multiple secondary participants.

3.2.4.2. Agreement

An agreement is made between the primary and secondary participants. This agreement is central to the process of disclosure as it states the terms on which information is released, thus communicating the subject's privacy preferences. This allows a subject to specify what use of their information they consent to when released. It also enables the subject to state any limitations or conditions associated with the disclosure. For example an agreement may state that the disclosed activity information may be used to assist in effectively managing incoming calls but that the information may not be further used or forwarded. The form and details of such an agreement will naturally vary greatly from situation to situation. In certain situations where very sensitive information is to be released the agreement can be explicitly stated in high detail, e.g. within a legally binding contract. In other day to day situations the agreement can be very relaxed and informal, in which case it may be implicitly determined by such factors as cultural and social norms, existing relationships, etc. Thus, with an agreement there is some capability of affecting the use of released information. One may even say it provides some degree of control. There are however no guarantees that the recipients of information will honour their agreement.

3.2.4.3. Trust

An important factor in determining whether or not to release information will therefore be the recipient's trustworthiness. Once a subject has established an agreement with the potential recipients, they need to evaluate if the recipients can be trusted to honour the agreement. It is only if the potential recipients are found to be trustworthy enough that information is released. This evaluation of trustworthiness adds an incentive for recipients to honour their agreement because if they fail to do so, their trustworthiness will undoubtedly be reduced and with that their future chances of receiving information. There is therefore a linkage between current and previous actions within the conceptual model. This obviously affects those who want to remain anonymous or that frequently change pseudonyms by introducing a trade-off between being anonymous and being trusted. Trust management is therefore not only essential for primary participants but also for secondary. All participants must be careful in deciding on who they trust as well as how they build up and retain their own trustworthiness.

Research into trust management and trust formation is, however, outside the scope of this work. For the purposes of this conceptual context model it is assumed these actions are carried out by the participants.

3.2.4.4. Ownership and disclosure

It should also be noted that even when a recipient is deemed trustworthy and information is released the ownership of information stays with the subject. The reason for this is that, unless anonymous, the personal information will still be linked to the subject after it has changed hands. Thus, by definition, the subject is the owner and should be in control over its flow. Disclosing information is therefore considered to be the equivalent of licensing its use rather than a traditional transfer of ownership. As such the process of disclosure is a business transaction in which the use of personal information is licensed to others.

Finally even though ownership is not transferred with a disclosure, information is for most other purposes in this conceptual model regarded just as any other good owned by the subject. Hence personal information, and its license for use, can be given away, traded, and even sold. Although this is not an area in which

this work will focus it is important that such behaviour is accommodated in the underlying model.

3.2.5. Legislation and social norms

In the last section it was stated that there are no guarantees of whether a recipient will follow their agreement or not. The only incentive presented for recipients to follow the agreements they make was that this will retain and possibly improve their future trustworthiness. There are however further incentives for a recipient to hold their end of the bargain such as legislation, social norms, and reputation.

In this model disclosure only follows after the terms of its releases have been agreed. This use of an agreement opens up the possibilities to further strengthen the privacy protection. An agreement can, as previously described, be explicit and stated in high detail within a legally binding contract. Such a document could also include the possible fines that will be incurred if the agreement is broken. In this way strong incentives can be ensured. The use of a dedicated contract is however not feasible in most situations. Not only is it time consuming and complicated to make an accurate agreement, but the process of being compensated once broken will most likely be long and expensive. In certain situations when dealing with very sensitive information, it may be an alternative.

Existing legislation that regulates the use of personal information, e.g. like the data protection act [Stationery Office 1998] in the UK, may also provide some protection. However, the applicability and application of such laws to ubiquitous computing environments must be further investigated before they can be relied upon. Furthermore, even if current data protection laws are applicable the problem of pursuing compensation remains. It is deemed, though, that legislation can act as a strong deterrent.

The social norms in society provide another incentive to respect people's privacy. They give guidance to what is found acceptable, and what is not, in our society. For instance it is generally acceptable to deduce information from occasional personal observations in public spaces. However it is not acceptable

to systematically observe a subject. This could be considered to be a form of surveillance or stalking. Similarly there exists a degree of tolerance for gossip, though depending on the circumstance it could be considered to be slander. Given that the breach of such social norms can make the offender an outcast in society, the potential is there to provide a reasonable deterrent.

Finally, the possibility of receiving a bad reputation can also act as a deterrent. This is a technique employed by sites such as eBay [eBay 2005]. The idea is that your actions, good or bad, are aggregated to form a reputation. This reputation can then act as a mechanism for establishing the trustworthiness of an entity. Hence it is desirable to retain a good reputation, which requires a subject to keep to their agreements. The application of reputation based privacy protection to ubiquitous computing environments is currently being researched [Goecks, Mynatt 2002].

3.2.6. Visualising the model

To visualise the model the idea that each subject has their own personal space is used. This personal space is where subjects keep information about themselves, which makes the ownership of this information clear. As the subject alone controls their disclosures and receptiveness, access to their personal space is limited to themselves. Thus, the release of information can only occur at the discretion of the subject.

The idea of personal space does not conflict with the notion of uncontrolled leakages, although it may appear to do so. A leakage is information that is deduced from the presence or actions of a subject. Hence, no information leaves the personal spaces.

As previously stated, only information regarding the subject is held within their personal space. By definition, the privacy of a subject only depends on them controlling the flow of their own personal information. Thus, information about others is therefore treated separately. In this privacy model, the use of others' personal information is governed by the agreement upon which it was released.

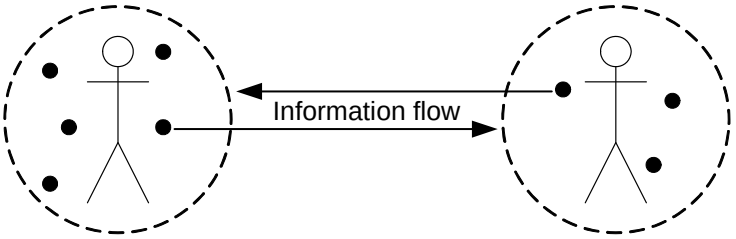


Figure 13. A visual representation of the personal spaces

Figure 13 shows two subjects, each with their own personal space and information. Although a subject is represented as an individual the model is not limited to this. Not only does the adopted definition state that also groups and institutions should be able to control their flow of information but this categorisation has been further relaxed to include anything about which information can be communicated. Also it should be noted that even though the personal space is represented as an area surrounding the subject, such space may be distributed provided the subject is in control. Finally the arrows represent the information flows that occur with the release of information. In this diagram both subjects have released information to each other.

3.2.7. Application to Context Model

So how is this model of privacy applied to the previously presented context model? To start with, the entities in the context model are the subjects. It is about them that information is held and communicated. Thus each entity has their own personal space in which their contextual information is held. As asserted previously the information must be about the subject, thus in this work ownership can only be claimed over information with respect to the particular relation that exists between an entity and a piece of context. This is important as it ensures that there is no conflict over the ownership of information in situations where a piece of context is shared between multiple entities as each entity will have their own relation to the information. The consequence of this mapping between the context and privacy models is that the control over context information should be with each entity. In cases where the entity is not an individual, an administrator can act on the entity's behalf.

3.3. Using the models

The models presented describe the position taken in this work with respect to context and privacy. This section aims to further clarify the use of the models with the help of a scenario. In particular the focus is on discussing various design issues and their consequences.

3.3.1. Scenario

Think of a typical office environment. There is a reception, offices, meeting rooms, coffee area, printer rooms, etc. Employees are situated in the various rooms around the building. Resources available are also in the form of computers, printers, telephones, photocopiers, stationary, etc.

Assume this environment has been equipped with a sensor network capable of accurately locating tags, worn by employees, within the building.

3.3.2. Entities and subjects

The context model does not place any limitation on what an entity must be (See above, section 3.1.1). Instead it encourages the use of narrower application specific definitions, as appropriate, in conjunction with the model. A key task when applying the model is therefore to further specify what an entity is.

Whilst anything in the above scenario could theoretically be defined to be an entity, it is in practise best to use things of interest as entities. In this case the employees are the key candidates. It is about them that location information is captured using the sensor network and tags. However, the rooms in the building can also be argued to be entities. The information captured by the sensor network applies to them as well. To understand the difference between these two alternatives assume that Bob, a worker, is present in the coffee area and then consider the following two statements: 1) Bob is currently located in the coffee area. 2) The coffee area is currently used by Bob. Both statements are equally valid but the information is structured differently. In the former statement the employees are selected as entities and in the latter statement the rooms are selected as entities. Whilst the structure has no impact on the information captured by the context model, it significantly affects the privacy model.

The entities in the context model are the subjects in the privacy model; it is about them contextual information is held and communicated (See above, section 3.2.7). The definition of what an entity is therefore also specifies ownership of information, which in turn has consequences for privacy control. For example, if in the above scenario the employees are chosen to be the entities then direct control over the disclosure of positioning information will be with the individuals. If, on the other hand, the rooms are chosen to be the entities then the control over the information will be with the administrator acting on behalf of the rooms. Consequently, from the employees' point of view, the information captured by the system will be an involuntary leakage.

Considering that it is the individuals that have a desire for privacy, it makes the most sense in the above scenario to define the employees to be the entities, and thus the subjects, in the model. However, there are situations in which it can be convenient, from an implementation perspective, to also define rooms as entities. For example, if the usage of a meeting room is to be monitored then defining it as an entity allows the system to retrieve who is located within it without having to query all the individual employees. How privacy control is handled when context information is shared is discussed later (See below, section 3.3.4).

3.3.3. Context elements

The definition of context is also an area in which the context model is general by design (See above, section 3.1.1). Again, the model does not place any limitation on what type of information constitutes context. For a data value to be recognised as contextual information the model only requires it to have a relation to an entity. Application specific definitions are then used in conjunction with the model, as necessary, to further specify what context is.

Assuming that the employees in the above scenario are chosen to be entities then any piece of information with a relation to the employees can be considered context. This produces an arbitrarily large context network. Hence, to make the context model manageable further restrictions must be introduced. For example, in this case it is possible to limit contextual information to be positioning data. Whilst this arguably makes the system location-aware rather

than context-aware, it is the only contextual information that has been specified to be of interest in the scenario. In practise, more types of information are generally of interest and a broader restriction must therefore be applied. For example, the definition presented by Dey and Abowd [Dey, Abowd 2000A] can be used.

A consequence of employing a narrower application specific definition of context is a reduction in the number of context elements supported. Whilst this is necessary to implement the model, it limits the types of applications that can be supported. For example, suppose the definition of context is limited to positioning information then the development of applications that require knowledge of the user's task is hindered. However, from a privacy perspective this specialisation can be argued to be beneficial. A reduction in the number of context elements that are supported implies that less information is held and communicated about the subjects.

3.3.4. Shared context

The context model allows situations to exist where a piece of context is shared between multiple entities. This occurs when there is more than one relation to a context item. As expected, this has consequences for privacy control.

Consider a situation where a meeting is taking place. Assume the sensor network, installed in the building, is able to capture accurately who is present at the meeting. On this occasion this includes several employees. Furthermore, for the purpose of the example, suppose there are only two pieces of context available: current location and meeting participants. In this situation, who owns the context information and who should be able to control the flow of information?

The privacy model states that the subjects have the right to control the flow of information about them (See above, section 3.2.1). It also specifies that it is with them that the ownership of information rests. When applied together with the context model it is further stated that ownership, and therefore also control, can only be claimed over information with respect to the particular relation that exists between an entity and a piece of context (See above, section 3.2.7).

Hence, in the situation outlined, each employee present at the meeting owns and has the right to control the disclosure of their current location. Furthermore, each employee also has the right to claim ownership and control over the context element meeting participants. However, since all employees have a relation to this element the ownership and control will not be exclusive, i.e. each employee can choose to disclose this information at their own discretion.

In situations where context information is shared, privacy control is significantly reduced. Since conflicts of interests can occur, a subject should treat shared context information as the result of an involuntary leakage over which they no longer have any control. Privacy control must therefore be exercised prior to any leakage of information and is limited to controlling their presence or recognition and to controlling their actions (See above, section 3.2.3). For example, in the situation above an employee can decide not to attend the meeting or to forget their tag, making it appear as if they are somewhere else, to minimise the leakage of information.

3.3.5. Implementation

Together the context model and privacy model lay the foundation for the development of privacy-friendly context-aware systems. The models identify the conceptual issues that exist and provide guidance on how conflicts should be resolved. It is then the responsibility of the implementation to ensure that the models are upheld.

The role of the infrastructure is to provide a system independent implementation of the models. This serves two purposes. Firstly, it enables a single implementation of the models to be used by any number of systems. This reduces the risk of the models being interpreted differently by separate systems. It also avoids work being duplicated. Secondly, it allows sensor networks and applications to be developed independently from each other and without knowledge of the underlying principles. Instead it is enough to follow the interfaces and protocols defined by the infrastructure. This simplifies the task of the developer.

As shown, because the models are general by design, a situation can be modelled differently depending on the decisions a system designer chooses to make. Whilst this can significantly affect the models, the consequences for the implementation are limited. Firstly, the selection of entities only determines who exercises control over the infrastructure components. For example, it does not matter if, in the outlined scenario, the employees or the rooms are chosen to be entities. The functionality required in the infrastructure, and thus the implementation, will be the same. However, differences can be expected in how the infrastructure is deployed and in the privacy preferences. Secondly, the adoption of a narrower application specific definition of context only affects the information handled in the infrastructure. To adhere to the context model the implementation must allow anything to be considered as context, provided a relation exists with the entity. For example, the implementation cannot be limited to handle only positioning information, as used in the above scenario, but must handle information in general. Hence, further specifying what context is does not affect the implementation of the infrastructure. On the contrary the application of a narrower definition during use can alleviate the issues that otherwise can occur if the implementation or the device(s) the infrastructure is deployed on cannot handle the demands of the context model.

3.4. Summary

In this chapter the definitions employed of privacy and context have been presented along with their associated conceptual models.

The definition of context presented state that "*context is information related to an entity, where the information may be an entity itself*" [Osbakk, Ryan 2003]. The emphasis is on the existence of a relation between entities and context. A piece of information can therefore be considered to be context in certain situations but not others. Furthermore, the definition does not restrict what can be considered to be an entity or a piece of context. Both these features have been shown to be important.

Based on the definition of context a conceptual model has been developed that represents context as a network of relationships between entities and context items. As demonstrated, it is not always easy to identify what is part of an

entity's context. The use of this model however, allows contextual relationships to be made clearer. It has also been described how the modelling of real world scenarios can be done, even though this may at first seem unreasonable due to the large quantity of relationships. This involves limiting the length of the relationship. Although there are issues concerning the optimisation of such limits, when used in application scenarios privacy will in practise provide a natural limit. Furthermore it has been shown how RDF can be used to represent the context model in a non-graphical format readable by both man and machine. RDF with its subject, predicate, object statements closely matches the idea of relationships used in the context model.

In this work privacy is viewed from the perspective of information flow. Westin's definition of privacy has therefore been adopted. It states that "*Privacy is the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others*" [Westin 1970 (p.7)]. This definition emphasises a subject's right to control the flow of information. Hence, privacy requires control not seclusion.

A conceptual model of privacy has therefore been presented that focuses on controlling the flow of information. In this model the position taken is that the introduction of context-aware ubiquitous systems must not adversely affect a subject's privacy, rather than aiming for perfect privacy. Hence, the ideal level of privacy is set to be that enjoyed offline. Two important assumptions are also made about the scope of control. Firstly, information constantly leaks from a subject's presence and action in public spaces. Secondly, control over information can only be exerted up to the point of disclosure. Thus in the employed model of privacy a subject's control is in practise limited to controlling their own disclosures, controlling their presence and recognition in public spaces, controlling their actions and links therewith in public spaces, and controlling their receptiveness. Another issue described is the process of disclosure. It is thought of as a business transaction where the use of information is licensed under an explicit or implicit agreement. The trustworthiness of a user and their intention to fulfil the agreement is thus an important factor in determining whether to release information or not. Social

norms, reputation, and perhaps legislation, have been presented as potential deterrents to the breach of agreements governing the release of information.

To clarify the use of models, a scenario has been outlined and used to discuss various design issues and their consequences. This has shown how the selection of entities in the context model is linked to the subjects in the privacy model and how this affects privacy control. Narrower application-specific definitions of context can be used in conjunction with the context model to control what information is included in the model. An example of a situation where context is shared has been presented and ownership and privacy control in it explained. Finally, the role of the infrastructure has been described and the consequences of the design decisions on the implementation discussed.

In the next chapter a privacy enhancing infrastructure is presented based upon the conceptual models of context and privacy as well as the background and project directives presented in previous chapters.

CHAPTER 4

THE INFRASTRUCTURE

In chapter 2 it was stated that the approach taken in this work would be experimental. With the conceptual models in place, the applied work can be presented. As previously described this involves the development of a privacy-enhancing infrastructure. The aim of the infrastructure is to improve both the level of privacy enjoyed by users and the support available for application development.

This chapter describes the developed infrastructure. It presents the requirements of the infrastructure along with the strategy and the scope of the design. Furthermore the infrastructure architecture is described together with some alternatives for the privacy protection. Finally the format used for context communication in the infrastructure is presented.

4.1. Requirements

In the previous chapters the problem, namely that of privacy in context-aware environments, has been described and the areas in question defined. This has corresponded to the problem phase in the research process. The next step is the requirements phase. It is within this phase that the requirements of a privacy-friendly context-aware infrastructure are captured.

Even though the area is fairly specific, being the intersection of privacy, context-awareness, and ubiquitous computing, this phase is still of the highest importance. The boundaries provided by the joint nature of the areas are themselves not enough to counteract the centripetal forces pulling the work away from its focus. It would otherwise be all too easy to go astray. Central to this process will of course be the context and privacy models presented as well

as the problem description. Indeed several requirements will be drawn from the models.

4.1.1. Capture process

Generally the requirement capture process (see Figure 14) entails finding, specifying, and validating all of the requirements of a project in detail. However, being a research project, the aim of the requirement capture has been slightly different. Instead of emphasising the need for a complete requirement-set, the process has focused on identifying the key issues the work should address. As such the requirement-set has been the stepping stones from which the problem space has been explored rather than being an absolute list of tasks that must be fulfilled. This has ensured the freedom necessary to research the area as well as keeping the research on track.

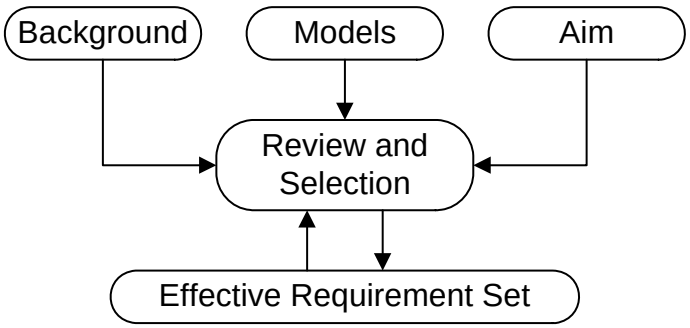


Figure 14. Requirement capture process

The process used to capture the requirements, i.e. the key issues, can be divided into two stages.

The first stage of the process has involved extracting potential requirements from various sources including the background, the models employed, and the work's focus. The 'background' source refers to the previous work done in related areas. This has been the starting point of the requirement capture process. By studying the results from related work a set of requirements has been built up. This includes both the features that related work has incorporated as well as those felt to be missing. The 'models' source refers to the adopted conceptual context and privacy models. This source has been especially important as the work needs to reflect the requirements stipulated by the underlying models. Thus by analysing the models and what they specify it has

been possible to extract and add further requirements to the set. Finally the 'aim' source refers to the work's focus and contribution. As such it groups together miscellaneous requirements that the work's focus as well as the intended area of contribution brings out. These requirements have been extracted from what has been set out as the aim of the work.

The second stage of the process has consisted of reviewing the set of requirements collected from the sources in the first stage and selecting which ones should be part of the effective requirement-set. The effective requirement-set is what later will be used in the proposal, experimentation, and evaluation phases. The review and selection of requirements has been performed iteratively during the requirements capture.

4.1.2. Captured requirements

This section will present the captured requirements set. To improve the overview the set will be broken down into three categories: privacy, functional, and miscellaneous. Under each category the relevant requirements will be described. It will also be indicated from which source the requirements have been captured. The sources possible are: [B]ackground, [M]odels, and [A]im (See above, section 4.1.1). In the cases where the requirements originate from multiple sources, all will be listed.

4.1.2.1. Privacy

The privacy category includes both those requirements that directly and indirectly affect a subject's privacy. Five different requirements have been captured, see Figure 15.

Requirement title:	**Source**
Retain offline level of privacy	A, M
Customisable effective balance of privacy	M
Handle known and unknown recipients	B, M
Decentralised structure	B, M
Security	A, B

Figure 15. Privacy-requirements

Retain offline level of privacy

It is required that a ubiquitous computing environment provides a subject with the same level of privacy as they currently enjoy in the offline world. Hence the infrastructure must not be more intrusive than the everyday environment in which we work and live.

This requirement initially originates from the aim source. It is the key motivation for this work to improve the overall level of privacy protection in context-aware ubiquitous systems (See above, section 2.1). The privacy model then formalises this aim and establishes the ideal level of privacy to be equal to that enjoyed offline (See above, section 3.2.2).

Customisable effective balance of privacy

It is required that the effective balance of privacy is customisable in the infrastructure. To reflect the scope of control the customisable privacy protection required can be broken down into three parts: access control, anonymity, and pseudonymity. Firstly an access control mechanism is required to enable a subject to have control over their own disclosures and to enable a subject to control their receptiveness. The mechanism must allow a subject to express their preferences with respect to their own disclosures and what information is solicited from whom. Furthermore the access control mechanism must enforce these preferences appropriately. Secondly a mechanism for achieving anonymity is required to allow a subject to control when they can be recognised and not in public spaces. Thirdly it is also required that a mechanism is in place that allows a subject to identify themselves using a pseudonym. This requirement goes further than simply being anonymous. It requires that a user is identifiable but not linkable to their chosen identity.

This requirement originates from the models source. First of all, the adopted definition of privacy makes the subject's control over the flow of information about them the focus. Secondly, the derived model of privacy then specifies what is considered the ideal level and the types of mechanisms that are required to achieve it.

Known and unknown recipients

It is required that the infrastructure supports both previously known and unknown participants. Hence a mechanism is needed with which a subject can express their privacy preferences independent of the identity of a recipient. Furthermore it must be possible for an agreement to be formed with previously unknown recipients.

This requirement originates from the background and models source. Firstly, the nature of ubiquitous computing is such that it is not feasible for a subject to know all participants in advance, there are too many of them. Secondly, the privacy model does not differentiate between previously known and unknown participants; a subject's privacy should therefore be enforced in both cases.

Decentralised structure

It is required that the infrastructure is decentralised and distributed. Subjects must not be required to have their personal information stored or processed by any centralised authority not under their control. Instead the subjects must themselves be allowed to decide where information is stored and processed. Furthermore the infrastructure must not rely on any single point for its operation, but should be able to function in isolation if required.

This requirement originates from the background and models source. Firstly the vision of ubiquitous computing speaks of the deployment of large numbers of devices in our environment [Weiser 2002]. This makes a centralised infrastructure impractical in terms of scalability. Secondly, the employed privacy model speaks of a level of control that is not achievable if a centralised infrastructure is forced upon the subjects (See above, section 3.2.3).

Security

It is required that adequate security is provided in the infrastructure. The infrastructure must therefore provide mechanisms to support authentication, confidentiality, integrity, availability, and anonymity. An adequate level of security is considered to have been achieved when a significant effort and considerable resources are required to breach the system. Hence, the system should be protected against attacks from the majority of users.

This requirement originates from the aim source. It is the aim to provide a privacy enhancing infrastructure. For this to be achieved security must be provided. The aspect important for security has then been drawn for background sources and Stajano's book on ubiquitous computing security [Stajano 2002].

4.1.2.2. Functional

The functional category includes the features found necessary to provide the basic functionality needed to support ubiquitous context-aware applications. Five basic requirements have been captured, see Figure 16.

Requirement title:	**Source**
Storage of context	B, M
Communication of context	A, B
Synchronisation of data	B
Multiple concurrent agents	B
Triggered actions	B

Figure 16. Functional-requirements

Storage of context

It is required that the infrastructure is able to store a subject's context. This minimises the duplication of data by applications. Context information should also be accessible at times other than when captured. Hence, the storage must support persistency. The storage must also be separate from the application and under the control of the subject. Furthermore, the type of data storage that can be used must not be restricted. This is essential to make best use of the, often limited, resources in ubiquitous computing environments.

This requirement has been captured from the background and models source. Many of the context-aware applications previously developed rely on contextual information being available rather than constantly sensed. The applications also utilise a wide variety of devices with different capabilities, many with constraints on the available resources. Furthermore, the privacy model specifies that a subject's context is held in their personal space.

Communication of context

It is required that contextual information can be communicated within the infrastructure. The communication must not be restricted to any particular medium or protocol. Instead the infrastructure should allow the necessary

support to be added when necessary. Hence, a standardised way of providing communication support must therefore also exist. Furthermore the infrastructure must support unreliable forms of communication. Finally with respect to the applications the actual communication must be transparent.

This requirement has been captured from the aim and background sources. A basic aim of the infrastructure is to communicate contextual information in a non-intrusive manner. Related work has also shown it to be desirable to separate applications from sensors [Dey 2000B]. Furthermore, ubiquitous computing is a large domain with a wide variety of heterogeneous devices being used. Thus it is not feasible to assume that a single form of communication can be used nor that connectivity is ever-present.

Synchronisation

It is required that the synchronisation of context information is supported in the infrastructure. A subject must be able to synchronise the contextual information held about them by one distributed component with another.

This requirement has been captured from the background source. The devices used in ubiquitous systems seldom feature a constant and reliable connection. This makes it difficult to guarantee that information can be updated in real time. Hence synchronisation at times of connectivity is essential, a feature common in mobile devices supporting Personal Information Management.

Multiple concurrent agents

It is required that support is provided for multiple concurrent agents in the infrastructure. Each subject must be able to utilise several context consumers and context producers simultaneously. Furthermore both local and remote agents must be supported. Finally the infrastructure must allow the set of agents to change at runtime.

This requirement has been captured from the background source. The related work demonstrates that contextual information has many uses, uses that cannot be excluded from coinciding. Previous applications developed also show that both locally and remotely sensed data is used in ubiquitous computing. Furthermore the users of ubiquitous systems are seldom stationary, but move

about. This causes users to often change environments and with them also agents.

Triggered actions

It is required that the infrastructure handles two types of triggered actions. Firstly the infrastructure must be able to handle context triggered events. Hence contextual changes must be detectable and the infrastructure must be possible to set up to act upon them. Secondly, the infrastructure must also be able to handle request triggered context collection. Hence the infrastructure must be able to collect information from context producers in a just-in-time manner.

This requirement has been captured from the background source. Being able to detect and react to changes in the environment is a central aspect of context-aware computing. Furthermore given the often limited resources of ubiquitous computing devices, the ability to reduce the frequency of context collection is beneficial.

4.1.2.3. Miscellaneous

Finally the miscellaneous category includes requirements for extended functionality. Three desirable requirements have been selected, see Figure 17.

Requirement title:	**Source**
Development support	A
Standalone and integrated operation	A
Interoperability with third-party systems	A

Figure 17. Miscellaneous-requirements

Development support

It is required that support is provided for easy and rapid application development. Privacy-friendly applications must not be more difficult to develop than those ignoring privacy issues. Hence, application developers must not be required to manage the privacy protection. Their responsibility should only be to publish the application's privacy practices, allowing the subjects to decide whether to release information or not.

This requirement has been captured from the aim source. It is an ambition of this work to improve the development support for privacy-friendly applications.

Indeed the hypothesis intended to be tested is whether the use of an infrastructure can help.

Standalone and integrated operation

It is required that the infrastructure can be operated both in a standalone and an integrated mode. Hence, infrastructure components must be able to be executed and run as separate processes. Furthermore, they must also be able to be embedded in other applications.

This requirement has been captured from the aim source. It is an ambition to provide as flexible development support as possible. Developers should be able to choose the mode of operation that is most appropriate given the requirements of the situation.

Interoperability with third-party systems

It is required that the infrastructure can interoperate with third-party systems. Contextual information must be able to be delivered and retrieved from systems developed by third-parties.

This requirement has been captured from the aim source. It is a goal that existing context-aware systems should be able to be utilised. The use of already deployed sensor networks is particularly beneficial. By interoperating with the different systems encountered by a user on a day-to-day basis, it is also hoped that these systems can be bridged.

4.2. Strategy

The goal of the infrastructure is of course to fulfil the requirements (See above, section 4.1). There are however different ways in which this goal can be pursued and depending on which route is taken the outcome will differ.

In many situations there will also be a trade-off between the different requirements causing the fulfilment of one requirement to affect the possibility to meet another satisfactorily. For example in this work a trade-off exists between the requirements that concern privacy and those that specify functionality. The reason for this is that privacy requires, by definition (See above, section 3.2.1), that the subject has control over the flow of their personal information while context-awareness on the other hand relies on information

being available to collect and process. Thus a subject's claims for privacy may hinder functionality.

A strategy is therefore needed that describes how to resolve such conflicts when they occur. In this work the strategy has been to place the emphasis on privacy and flexibility. What this implies is discussed in the following sections.

4.2.1. Prioritise privacy

From the start the improvement of privacy has been both the driving force and the goal of this work (See above, section 2.1). Thus the position taken during the design and implementation of the infrastructure is that privacy is the most important objective.

It is especially important to be clear on this prioritisation as the effectiveness of the privacy protection depends on the infrastructure as a whole rather than individual parts. For example, if the infrastructure strictly controls the flow of information to individuals but takes a more relaxed approach to automated services then it is likely that suspect individuals will tunnel their requests through dummy services. Thus the real level of protection offered to a subject is considered to be that of the lowest denominator. This means that privacy always needs to be prioritised so as to not void the protection. Hence, any functionality must be weighed against its potential impact on privacy and if necessary be adjusted or counteracted before it can be incorporated into the infrastructure. Take for example the functional requirements "Communication of context" and "Multiple concurrent agents" presented in the previous section (See above, section 4.1). Together these require that remote agents should be able to retrieve a subject's contextual information. Unmodified, this would be a direct violation of a subject's privacy. The privacy requirements must therefore take precedence here, allowing suitable protection to be put in place even if this may restrict the desired functionality.

Furthermore, the prioritisation of privacy must be applied from the beginning of the design process. It is at this early stage the real opportunities exist in ensuring that a privacy-friendly product is attained. Once the functionality has been decided and the design started, adding privacy would involve either redesigning

the product or wrapping it with the privacy protection. The former is of course time consuming and therefore also costly. The latter would generally interfere with the already designed functionality and may still even then not provide the desired overall protection. This was recognised already in the early work at PARC [Weiser 2002]. Privacy then needs to continue to be taken into account throughout the design process.

The strategy taken in this work has therefore been to prioritise privacy over any other functionality from start to end of the project. This implies that in situations where requirements diverge or conflicts occur, privacy always takes precedence.

By doing this the desire has been to maintain an adequate level of privacy throughout the infrastructure design, and also later in the implementation. It has also ensured that a minimum amount of time has been spent on redesigning parts of the infrastructure to fulfil the privacy requirements.

4.2.2. Modular design

It is also the objective of this work to improve the support available when developing context-aware applications. This is hoped to be achieved by moving much of the responsibility from the developer onto the infrastructure. Whilst this is obviously beneficial when it comes to simplifying the tasks a developer needs to perform, it does however limit their freedom.

From the requirements presented in the preceding section (See above, section 4.1) it is clear that a single solution will not be able to encompass every scenario. The heterogeneity of the target devices, and their capabilities, means that many different solutions need to be supported. For example it is required that no limitations are placed on either the data store or the forms of communication that can be used (See above, section 4.1.2.2). To be able to fulfil these and other conditions, the infrastructure must be flexible.

In contrast with the approach taken in previous work such as the Context Toolkit [Dey, Abowd 2000C], where the toolkit itself was extendable [Dey 2000B s.6.], the strategy taken in this work to provide the required flexibility has been to make the infrastructure modular wherever possible. By using modules the behaviour of one part of the infrastructure can be optimised or

customised without requiring changes to be made to other parts. This is important in order to retain the unity of the infrastructure. Furthermore, by controlling which parts of the infrastructure are implemented as modules and what functionality they encompass, the overall behaviour can be guaranteed. From a privacy perspective this is important as even small changes to the handling of information may result in serious loss of privacy. The modular approach also preserves a high level of abstraction which is believed to be beneficial when ease of development is desired. By making ready modules available for common configurations a majority of developers and users should be able to put together a suitable setup from existing parts. In this way they simply need to know what module or functionality is required, not how it is implemented. For example, there may be one data storage module available that uses a file for storage and another that uses a database.

4.3. Scope

Designing an infrastructure for context-awareness even for the requirements presented in the previous section (See above, section 4.1) is an ambitious task. With the focus on ubiquitous computing environments come limitations. Resources such as processing power, memory, data storage, and battery power are scarce compared with those available in traditional computing environments. This is the case even when working with relatively powerful devices. Thus, there are restrictions on what can be achieved as well as a need to be extra careful with resources. The wide range of devices used in ubiquitous computing environments, and the lack of common ground between them, also has its implications. Each type of device comes with its own set of features and limitations. This is of course problematic when developing a uniform infrastructure. The more diversity that exists, the harder it is to achieve an all-inclusive solution.

While the impact of these aspects can be seen throughout the work, two areas of the design are affected more directly: the context model and the hardware platform.

4.3.1. Restricting the context model

Earlier in this thesis (See above, section 3.1.1) context was defined as information related to an entity, where the information may be an entity itself. The emphasis is on the existence of relationships, rather than specific characteristics. Thus no limitations are placed within this definition on what can and cannot be context. Instead the condition for when something is to be considered context is specified. While this generality is desirable it does complicate things.

Because of the emphasis on relationships the contextual model has a network structure (See above, section 3.1.2). When applied to the real world the network will be almost infinitely complex since relationships exist between everything around us. As shown previously even between seemingly unrelated entities a relationship-chain can be established (See above, section 3.1.2). Thus the context graph will from any given point be several levels deep and may also contain relationship loops. This makes the model difficult and perhaps even impossible to create, let alone process, without further limitations. It is therefore necessary to reduce the complexity of the context model, to a level that is feasible to process.

The first step towards a more manageable model is to reduce the depth of the context graph. Each entity in the model, i.e. that is associated with context, must per definition be related to at least one entity. In reality they are however, likely to be related to many more. For example, a person may have three friends, and each of the friends may in turn have three friends of their own, and so on. Thus restricting the depth of the model will have a significant effect on the graph size, illustrated in Figure 18.

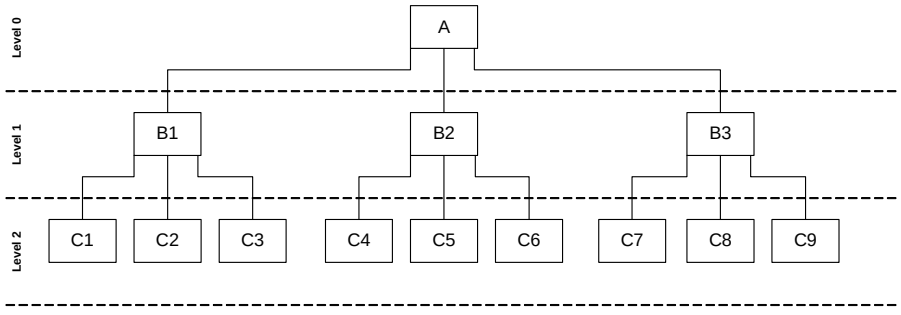


Figure 18. Growing context model

The degree to which the depth should be limited is in this work determined by the privacy model. The employed model of privacy (See above, section 3.2) states that each subject has their own personal space in which information about them is held. The context model is therefore in effect split into several sub-models, where each one represents the context of a particular entity. Furthermore, since the personal space only contains information about the subjects themselves, the depth of each sub-model is restricted to one level. Thus the size of the graph that needs to be handled for each entity is reduced.

However it is not enough to limit the depth of the context model. Even one entity's sub-model, with its depth limited, is likely to be too large to be handled effectively on a mobile device. For example, think of the number of direct relationships that exist between a person and their family, their friends, the places they go to, and the objects they interact with. Just the relationships that are 'active' on a daily basis can form a link with several hundreds or even thousands of entities. Even though the possibility to automatically establish what relationships exist provides a natural limitation, this limitation cannot be relied upon. As technology advances relationships will become easier to detect and any such limitations will gradually disappear.

The second step towards a more manageable model will therefore be to reduce the number of relationships. It is assumed that there will always be a subset of relationships that are found to be more interesting than others. The approach taken has therefore been to limit the context model to only contain relationships of interest which will reduce the size of the model. To what degree, will of course depend on how many relationships are found to be interesting. So what is interesting? This is determined individually by each entity (or its administrator) and will vary. As such an entity will be able to scale their context graph to fit the resources available by changing the size of the subset with relationships they find interesting.

By both limiting the depth of the context model and reducing the number of relationships it contains it is possible to attain a context model that is possible to process even on the limited devices that are used in ubiquitous computing environments.

4.3.2. Targeting a specific device type

Related work has shown there to be a wide variety of devices being used by ubiquitous computing systems (See above, section 2.6.3). This diversity is certainly beneficial when attempting to incorporate the devices into our environment as situations differ. However, it also means that there is no common platform on which systems can be built. This makes it difficult to develop a uniform infrastructure that can run on all the devices being used. Thus, it is necessary to target a specific subset.

4.3.2.1. Criteria

To decide upon what type of devices to focus on, a number of different aspects have been considered. Together they form a set of criteria that have been used to select the device type to target.

Capabilities - The device type needs to be able to fulfil the demands of the infrastructure. This includes providing enough processing power, memory, and storage to handle the flow of information. It also takes into account the available means of communication and interaction.

Versatility - The platform must also be flexible enough to work in many environments, to adapt to different situations, and to allow customisation with respect to user preferences. Thus it is important that the devices provide choices both in terms of the capabilities they offer and how they are configured.

Availability - The type of devices must be widely available. This means using common off-the-shelf devices whenever possible, a point which the work has focused on from the start (See above, section 2.2).

Cost - The type of devices selected must also be reasonably priced. It is important that the cost is such that it is indeed possible to mass deploy an infrastructure.

Ubiquitousness - Finally the chosen type of devices must be or have the potential of becoming pervasive. In line with the vision of ubiquitous computing [Weiser 2002] the devices should be both ever-present and seemingly invisible.

4.3.2.2. Personal digital assistants

For this infrastructure the choice fell on off-the-shelf PDA-type devices. The motivation behind this choice is as follows.

Firstly, in comparison with the other resource constrained devices, PDAs are quite powerful. It can even be argued that the limiting factor now is battery life rather than processing power and memory. Furthermore, PDAs come with many different features in a single package, e.g. colour display, sound input/output, text input/output, and various connectivity options including wireless.

Secondly, PDAs are versatile. Since they are mobile and have a small form factor, it is possible for them to be used both on the move and as part of a fixed infrastructure. They also offer further options for customisation through different models, accessories, and software configurations.

Thirdly, the availability of PDA-type devices is good. PDAs can be bought from most consumer electronic shops. There is also a large assortment of models available from several different manufacturers.

Fourthly, the cost of a PDA, whilst considerable, is still regarded to be reasonable given the features these multipurpose devices have.

Finally, PDAs exhibit the characteristics of a ubiquitous device. They are mobile devices that a user often brings with them. Furthermore, even though the device itself is visible, the interaction with PDAs is gradually becoming so natural as to appear invisible. They can be thought of as mobile personal servers in a ubiquitous computing environment.

Altogether these aspects show personal digital assistants to meet the criteria set up.

4.4. Architecture

The architecture of the infrastructure is closely tied to the employed conceptual models (See above, Chapter 3). It has also been designed with the presented strategy (See above, section 4.1) and scope (See above, section 4.3) in mind.

Overall the infrastructure is perhaps best described as being decentralised with islands of control. Each island is a centre of information that operates

independently within the infrastructure. These centres are referred to as context managers and they are the main components in the infrastructure.

In addition to the context managers there are also agents, data stores, and context manager catalogue services. All of these components interact within the infrastructure and will now be described together with the context manager.

4.4.1. Context managers

The context managers are the central components in the infrastructure. They are in charge of handling the contextual information.

4.4.1.1. Relation to entities

Each entity in the infrastructure is associated with at least one context manager. This is necessary to ensure a single level relationship chain as required by the depth restriction placed on the context model (See above, section 4.3.1). Note, it is possible for an entity to distribute their contextual information over several context managers if desired. A context manager on the other hand can only be associated with at most one entity. To better understand why this is the case we need to return to the conceptual model of privacy (See above, section 3.2) and look at how the context manager fits in.

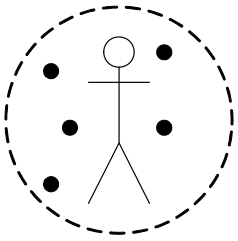


Figure 19. Entity and its personal space

In the model the idea of a personal space is used to represent the place where a subject holds information about themselves, illustrated in Figure 19. Access to this personal space was stated to be restricted to the subject alone. However an entity has a special relationship with its associated context manager(s). A context manager is not considered to be an external entity but an extension of the subject themselves. More precisely, the context manager sits on the boundary between the personal space and the surrounding public domain, as illustrated in Figure 20. This allows the context manager to act as an

information gateway. Since the context manager forms a part of a subject's personal space it cannot be associated with more than one entity.

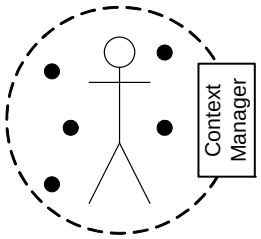


Figure 20. Entity and its personal space and context manager

4.4.1.2. Responsibilities

The context managers have been designed to provide much of the required functionality (See above, section 4.1) in the infrastructure. First of all a context manager is responsible for storing an entity's contextual information. This is the most fundamental duty and one that is necessary for the operation of the infrastructure. The context manager is also responsible for making the stored context information available. It is through an entity's context manager(s) that all their information flows. Related to this is of course another duty, namely that of privacy protection. Given that the flow of information passes through a context manager, it is there the privacy protection must be applied. The context manager must also provide the capability to synchronise data. Synchronisation is needed to make the use of multiple context managers feasible. The data that needs to be synchronised includes contextual information, privacy preferences, and settings. Finally it is the context manager's duty to ensure its address information is kept up-to-date.

4.4.1.3. Multiple context managers

It was stated above that an entity in the infrastructure can be associated with multiple context managers. The motivation for this is as follows. Firstly, as the infrastructure targets a ubiquitous computing environment the accessibility of the context managers is expected to vary. This could be because of limited or intermittent connectivity, power management, or simply device usage. Through the use of multiple context managers it is possible to improve accessibility, particularly if one instance is run on a server. Secondly, the infrastructure targets resource-constrained devices. This limits the rate at which requests can

be performed. By distributing the context information over multiple context managers the load can be shared and performance improved. Finally, associated with the use of small mobile devices is an increased risk of accidental damage or loss. It is therefore especially important that data is backed up regularly. The use of multiple context managers to duplicate data provides a convenient alternative.

4.4.2. Agents and other actors

Agents are the actors that allow the infrastructure to interact with the surrounding environment. It is they that communicate with the context manager to store and retrieve contextual information. Figure 21 illustrates the information flow between a context manager and an agent.

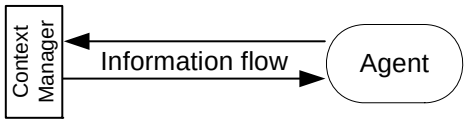


Figure 21. Information-flow

The infrastructure contains two types of agents, context producers and context consumers. The context producers collect contextual information about an entity and send it to their context manager. From the perspective of the context manager such agents appear as the source of the information. In most situations though, context producers will be middleware that enable information to be extracted from sensors or sensor networks. The context consumers on the other hand retrieve and use the information stored by context managers. Hence it is they that provide users with context-aware services. Context-aware applications will therefore be context consumers.

In addition to agents there are two other actors in the infrastructure that provide supporting services namely data stores and catalogue services. The former, data stores, provide context managers with the data storage they need. Hence although a context manager is responsible for the storage of contextual information it will only manage this process. How the information is then stored physically depends on the data storage used. With this separation the context manager becomes independent from any underlying storage mechanism allowing it to change as required (See above, section 4.1.2.2). The latter,

catalogue services, provide a service that allows context managers to publish how they can be contacted and also enables agents to retrieve such information. Thus a catalogue service makes it possible for a constant address to map to the dynamically changing contact details of a context manager. This functionality is necessary in the infrastructure to support unreliable communication over an arbitrary number of protocols and mediums (See above, section 4.1.2.2).

4.4.3. Component interactions

The interaction between the different infrastructure components plays a vital role in the infrastructure. There are altogether five different types of base components: context managers, context producers, context consumers, data stores, and catalogue services (CMCS). Each of these interacts with at least one other component in the infrastructure. Figure 22 gives an overview of the interactions that occur within the infrastructure. Note that the catalogue services interact with any component that needs to look up addresses or update address entries.

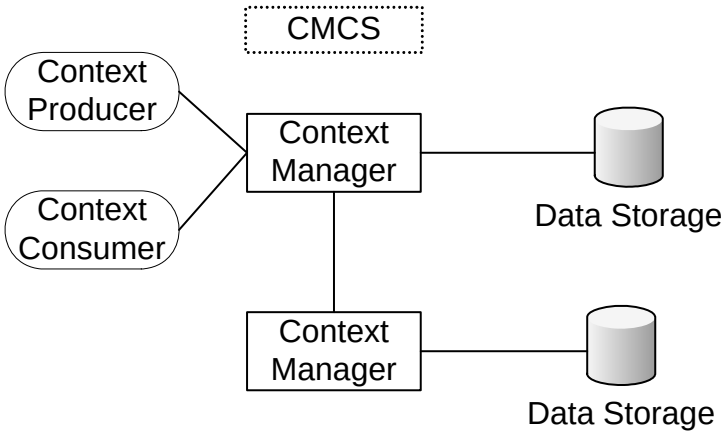


Figure 22. Infrastructure components

Since the majority of the interaction within the infrastructure will occur between a context manager and an actor the interaction will be presented from a context manager's perspective. From the responsibilities presented in the previous section (See above, section 4.4.1) six use cases have been identified: *store context*, *retrieve context*, *synchronise data*, *process request*, *handle catalogue service*, and *handle data storage*. All use cases except process request, which is internal, map onto a different infrastructure component, as shown in Figure 23. Note the separate use case for handling the data storage. This serves the purpose

of emphasising the design choice of modelling the data storage as a external actor.

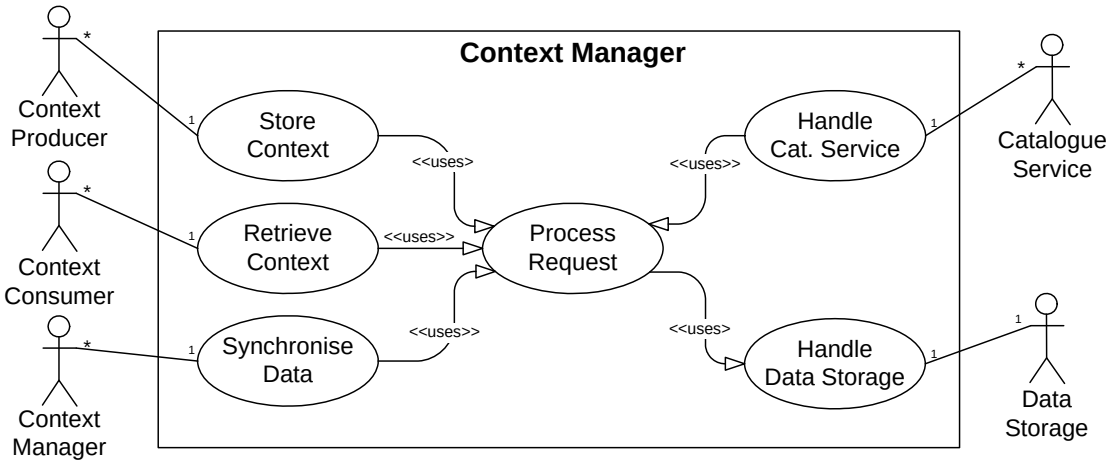


Figure 23. Use Cases

Store context - This use case enables the context managers to receive contextual information from various context producers in the infrastructure. The interaction between the context manager and the context producers occurs through one of two mechanisms, push or pull. The push-mechanism allows a context producer to push information to the context manager for storage. Hence the context producer initiates the interaction and transfer of information. The pull-mechanism on the other hand allows the context-manager to explicitly request information from a context producer. With this mechanism it is therefore the context-manager that initiates the information transfer. Regardless of whether the information is pushed or pulled, the context manager evaluates the information when received and if appropriate stores it.

Retrieve context - This use case allows contextual information to be delivered to context consumers. There are two mechanisms by which the context manager can achieve this, request and publish. The request based mechanism requires context-consumers to initiate the process by placing a request for the piece (or pieces) of information they require. Upon receiving such a request the context manager then evaluates the request and returns an appropriate response. In contrast, the publish-based mechanism is initiated by the context manager and is not restricted to delivering information only when requested but could do so at, for example, set intervals or when triggered by changes in context.

Synchronise data - This use case provides the means for context managers to synchronise their data with each other. The interaction between the context managers is request based and can be initiated by either context manager. The process involves three interactions. It starts with the initiator requesting a snapshot of the available data items, including identifiers and timestamps. This request is then dispatched to and processed by the opposite party. The opposite party then evaluates the request and adds identifiers and timestamps as appropriate before replying. Upon receiving the response the initiator evaluates it against the data items contained locally. Then depending on which type of synchronisation is being performed, local / remote / local & remote, data is either downloaded, uploaded, or downloaded and then uploaded.

Handle data storage - This use case concerns the interaction with the data storage. This interaction, whilst important, depends on the type of data storage employed. The interaction can therefore not be completely specified. From the perspective of the context manager however, the interaction must be seamless. Thus, even thought the interaction between the data storage and its handler may change, the interface towards the internal structure of the context manager will remain fixed.

Handle catalogue service - This use case allows context managers to interact with the catalogue service to update the address record. The interaction is request-based and is initialised by the context managers. Each time the address(es) to the context manager change(s), an update request is sent to the catalogue services subscribed to. The update request contains the current contact information allowing the catalogue service(s) to update the record held.

As can be seen from the interaction between the base components, the infrastructure is focused on performing particular tasks, for example the context producer only interacts with context managers to store information. This makes the interaction and the flow of information clearer. Thus while it is possible for an actor to take on multiple roles, it is not recommended.

4.5. Privacy protection

It is the main aim of this work to improve the available privacy protection and several requirements have therefore been presented (See above, section 4.1.2.1) on this topic. There are various mechanisms that can be used to fulfil these requirements. This section discusses some of the alternatives considered for use within the infrastructure to perform authentication, control access, and achieve anonymity/pseudonymity. The section also examines the concept of notice and how it is handled within the infrastructure.

4.5.1. Authentication

Authentication is an important part of the privacy protection, after all a key determinant in deciding whether to release information is often the identity of the recipient. In the authentication process it is irrelevant what this identity represents, but simply that the authenticity of the claim is determined.

Stajano outlines three methods for authentication [Stajano 2002]. Firstly, passwords can be used. This is the simplest of the three mechanisms. It only requires a common secret to be known by the involved parties. Hence, it can be used with very resource constrained devices. The disadvantages though, of a password-based authentication mechanism are that it is susceptible to replay attacks and can be difficult to manage. Secondly, a chain of linked one-time passwords can be used. This mechanism uses a verifiable sequence of single use passwords for authentication. In this way replay attack can be avoided. This gives rise to another disadvantage though, namely that both parties need to be synchronised. Here synchronised means that they need to be at the same index in the password list. Also every time the end of the list is reached a new sequence needs to be generated. Finally, a challenge-response based mechanism can be used. This type of mechanism establishes the identity of a user by posing a challenge and verifying the given response. This allows replay attacks to be avoided without any need to retain synchronisation between the parties. However, given that public key cryptography often is used with this type of mechanism it is more demanding in terms of the resources required. It is also susceptibility to man in the middle attacks. Man in the middle attacks can be

counteracted by verifying the origin of public keys using a trusted third party, assuming such a party exists and is available.

Which mechanism can and should then be used with respect to this infrastructure? The use of one time passwords can be excluded. The need to retain a synchronised state is not compatible with either the distributed architecture proposed or the requirement to support unreliable connectivity. Hence, the choice stands between using a password based mechanism or a challenge-response based mechanism. Both of these can technically be used in the infrastructure. The choice therefore depends on what should be prioritised: a challenge-response based mechanism is more secure, whilst a password based mechanism allows a wider selection of platforms to be supported.

4.5.2. Access Control

Access control is at the centre of the employed model of privacy (See above, section 3.2). It is via the access control mechanism that a subject is given the power to control their own disclosures and their receptiveness. Three different mechanisms have been investigated in this work. Due to the scope of the work all three focus on static preferences with respect to identity, which have been shown to be stronger [Lederer, Mankoff et al. 2003].

4.5.2.1. Classification and Clearance Scheme

In the early experiments [Osbakk, Ryan 2002] the use of a classification and clearance scheme to structure privacy preferences was evaluated. The mechanism, which will be referred to as CCS, captures privacy preferences in levels of sensitivity and trustworthiness. The underlying thought is that as one places greater trust in someone, more sensitive information can be revealed to them. This assumption forms the basis for this simplistic access control mechanism. Figure 24 illustrates the relationship between sensitivity and trustworthiness.

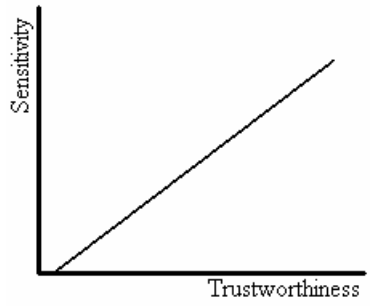


Figure 24. CCS trustworthiness and sensitivity

In CCS each context element is assigned a classification level, indicating its sensitivity. The more sensitive the information is the higher the level assigned. For experimental purposes a range between 0 (public) and 5 (private) was used in the evaluation, though there is no technical limitation hindering the use of a larger range. Context consumers are assigned a clearance level, indicating their trustworthiness. This clearance level then determines what information they can access. Only if the context consumer has a clearance that is higher or equal to the classification of a requested piece of information will it be released. The ability to store information is controlled by a context producer flag. Thus, only an agent that has been given producer 'status' can store information.

With only two types of variables being used, classification and clearance levels, the advantage of this mechanism is its simplicity. Take for instance the process of evaluating whether or not to release a piece of information. This becomes a straightforward comparison between the recipient's clearance level and the classification of the information. Furthermore, since neither the classifications nor the clearance levels change dynamically, this comparison can be made highly efficient through the use of cached lookup tables. Another aspect that needs to be considered is the procedure needed to set up the access control mechanism. With CCS this merely involves going through the list of context elements, classifying them according to their sensitivity as appropriate, then assigning clearance levels to context consumers with respect to how trustworthy they are. Since each context element and consumer is only associated with a single level the workload rises linearly with the increase in numbers.

The disadvantage of the approach is that it does not scale to deal with large numbers of context elements or consumers. The problem is that the CCS

mechanism relies on the assumption that the more trustworthy someone is the more sensitive information they should be able to retrieve. Hence, a more trushtworthy context consumer is always able to access a superset of the context elements that a less trustworthy context consumer can access. When the number of context elements and consumers increases, the probability that a subject's preferences will fit this assumption decreases. Any exceptions that occur will cause either too high or too low access to be granted. Consequently the expected accuracy with which privacy preferences can be expressed decreases. Consider the following scenario. Assume there are two context consumers, Alice and Bob, and two context elements, activity.note and location.place. Furthermore, assume that the subject feels that Alice is the more trustworthy context consumer and that activity.note is the more sensitive context element. Finally, assume that the subject wants Alice and Bob to be able to access activity.note and location.place respectively. Note that the subject does not want to disclose location.place to Alice despite her trustworthiness. The reason for such an exception could be that the knowledge Alice has about the subject allows her to infer more from the information. In this scenario a mismatch exists between the subject's privacy preferences and the capabilities of the CCS mechanism. Since neither of the context consumers' access can be expressed as a superset of the other's access the subject's preferences cannot be represented accurately. In such situations the subject must decide whether to release more or less information. For example, the subject can either accept that location.place is released to Alice or lower Alice's clearance level below the classification of location.place, consequently also denying Alice access to activity.note. Neither of these alternatives is of course desirable, but accepting the need to compromise allows the CCS mechanism to scale better. However, even selecting such compromises becomes increasingly difficult as the numbers of context elements and context consumers increases.

Whilst the CCS mechanism is straightforward to process and exhibits a linear configuration workload, the failure to accurately capture a subject's privacy preferences in certain situations, as demonstrated by the scenario above, cannot be ignored. The requirements specify that a subject should be able to control

their own disclosure of information (See above, section 4.1.2.1). While it is possible to improve upon the accuracy of the CCS mechanism by introducing individual classification lists, this would both complicate the processing and significantly increase the configuration workload. Thus for increased accuracy another mechanism is required.

4.5.2.2. Role Based Access Control

In response to the limitations found with CCS further experiments have been performed with Role Based Access Control (RBAC) [Sandhu, Coyne et al. 1996]. The idea behind RBAC is to use roles to group permissions together. This makes administration easier and provides scalability, by allowing permissions to be reused. Another important aspect is that the possible accuracy of the RBAC mechanism is not affected as the number of context consumers and context elements increases.

The role based mechanism evaluated in this work is based on the $RBAC_0$ model [Sandhu, Coyne et al. 1996]. As such there are three variables: agents, roles, and permissions. *Agents* are the entities that desire access to information, i.e. context consumers and context producers. *Roles* denote the functions held by users, associating them with certain rights and responsibilities. Finally *permissions* capture the consent to data access and are always positive. Together these variables allow a subject to describe their privacy preferences. This process involves defining permissions, assigning permissions to roles, and finally assigning roles to agents. Figure 25 illustrates the relationship between the variables.

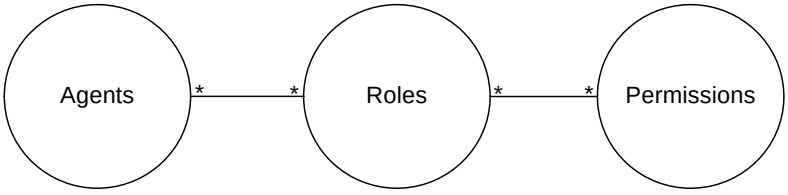


Figure 25. RBAC variables

However, while based on the $RBAC_0$ reference model, this mechanism has an important difference in that roles are automatically activated as required. This simplifies the use of the access control mechanism as agents do not need to decide what roles to invoke when requesting information. The result is that

agents, i.e. context consumers and context producers, will always be granted the best possible access given their current set of roles. For example, if Alice has been assigned two roles, friend and colleague, then her access, at any time, will be the permissions held by both roles. Assuming the roles friend and colleague grants access to user.home-info and user.business-info respectively then Alice can access both context elements.

The permissions have been implemented as lists of access controls. Each access control grants access to one context element. Any combination of read, write, and history access can be specified, including none of them. In the case of history access, a limit can be set on how far back access is granted. For further customisation the possibility of agents being assigned a personal permission has also been introduced. The personal permission *overrides* any permission granted by more general roles, allowing the access to be fine tuned. This capability helps to minimise the administrative burden of handling exceptions.

Compared to CCS the RBAC has an important advantage, scalability. Firstly, and most importantly, the accuracy of the RBAC mechanism is not affected as the number of context elements and consumers increases. The subject can represent their privacy preferences just as accurately with respect to a single agent and context element as to many. The reason for this is that preferences are expressed as exact permissions describing access in detail. While the grouping of these permissions onto roles may appear to reduce the exactness, they do not as the mapping of permissions onto roles is free. Furthermore, the introduction of a personal permission allows agent specific preferences to be expressed. Thus it is possible, if required, to control access at an individual level. Secondly the use of roles offers benefits in terms of administration. Whilst there can be many agents requiring access to contextual information, it is not uncommon for the number of different roles these agents take, with respect to the subject, to be limited. By expressing privacy preferences using roles, then reusing these for many agents, the workload of setting up the access control mechanism can be reduced. For example a person may have several friends, all of which should enjoy similar access. The use of a friends role allows the common permissions to be setup once and then reused for each friend, leaving only exceptions to be

expressed individually. Finally, the RBAC mechanism facilitates a loose coupling between agents and permissions. This allows an agent's access to progress, through role changes, without altering permissions. For example as a subject gets to know someone better they can improve their access by changing the assigned role from acquaintance to friend. Indeed Sandhu et al. [Sandhu, Coyne et al. 1996] point to evidence showing that changes to role membership is more frequent than changes to role permissions.

There are however also drawbacks to using the RBAC mechanism. Even though the reuse of roles helps to keep the administrative burden down, the RBAC mechanism requires a significant initial effort to be setup. Before any agents can be granted access, the appropriate roles and their associated permission need to be setup. This is a slow process, particular if the privacy preferences are complex, as permissions express privacy preferences in detail. Furthermore what roles a subject should define is not always clear, nor is the mapping of permissions onto roles. For example should a separate role for acquaintances and friends be used, or is it enough to have a single role with exceptions? The decisions made will directly affect the effort required to setup the mechanism, but also that of maintaining them later. Any strategies must therefore be carefully evaluated. Another disadvantage is that the access control mechanism is 'non-fluent', i.e. access is granted using distinct permissions, or groups thereof. This makes it difficult to evaluate whether a particular setup provides better access than another, unless one is the superset of the other. Thus, the process of improving an agents access is not as simple as increasing their clearance level. Instead an individual assessment needs to be made about which roles or permissions need to be added or changed.

Overall though, the RBAC mechanism is an improvement upon CCS. By providing a subject with facilities to control the access to their information in detail, the accuracy issue displayed by CCS has been addressed. The use of roles also reduces the effect of the increase in the administrative burden the more detailed control brings. However, there is still a significant effort needed to set up the mechanism initially. Furthermore, the non-fluency of the RBAC

mechanism hinders the subject from making small but general increases in access.

4.5.2.3. The Platform for Privacy Preferences (P3P)

Both the CCS and RBAC mechanisms express privacy preferences with respect to previously known agents. There are however many situations, particularly in ubiquitous computing, where the identity of agents cannot be known in advance. To handle these situations a supplementary mechanism based on the Platform for Privacy Preferences (P3P) [World Wide Web Consortium 2002A] has been evaluated. The mechanism exploits the fact that an agreement is made between a subject and an agent during the process of disclosure (See above, section 3.2.4), allowing access decisions to be made on the basis of agreed information usage.

The P3P based mechanism uses two types of data: P3P policies and rulesets. The former, the P3P policy, represents the agreement made between a subject and a recipient regarding the use of information. It is the contract that describes what the recipient may do once the information has been received. For example, the contract may state that the purpose of the data collection is to allow them to tailor the subject's visit. When using the P3P-based mechanism the contract is formulated by the requester alone, declaring their intentions. The latter, rulesets, represent a subject's privacy preferences. Each ruleset contains a list of rules and an access control mapping. The rules specify conditions that must or must not be true. For example a ruleset can contain two rules. One rule could state that data may be collected to allow visits to be tailored. Another may state that the collected data must not be stored after the end of the session. The access control mapping provides the link with the underlying access control mechanism being supplemented with P3P. Being privacy preferences, the rulesets are consequently defined by the subject.

To allow assertions to be made about contextual information, the P3P vocabulary has been extended. The extension adds several context specific data elements, see Figure 26. For example an element called location.place has been added to the vocabulary, allowing preferences and intentions to be describe with respect to location information. Though, whenever possible, existing P3P data

types have been reused. For instance, instead of defining a new data type for addresses, the postal data type is reused. For the mechanism to work both P3P policies and rulesets need to employ a compatible context vocabulary. Details of the vocabulary used can be found in Appendix A.

Data elements:
location.place
location.postal
location.coordinates
activity.note
activity.type

Figure 26. Examples of context vocabulary

With the P3P based mechanism a two-step process is used to resolve access, see Figure 27. In the first step the P3P policy is evaluated against any predefined rulesets. This involves checking whether the policy obeys the rules made by the subject. Each ruleset is evaluated independently and only a binary outcome is possible. Either the policy obeys *all* the rules in the set, in which case there is a match, or at least one rule is broken, resulting in a mismatch. Once the policy has been evaluated against all the rulesets and at least one match has occurred, the second step is carried out. The second step resolves the link between the matching rulesets and the underlying access control mechanism. Depending on which access control mechanism is being supplemented the details of this step varies. When used with CCS the rule sets maps onto clearance levels. Thus a successful match between a P3P policy and a ruleset grants the requestor the associated level of clearance. On the other hand when used with RBAC the rulesets map onto roles. Hence a successful match therefore assigns the requestor with the associated role or roles. Once the appropriate mappings have been made the underlying mechanism can resolve the access as if the requestor was known.

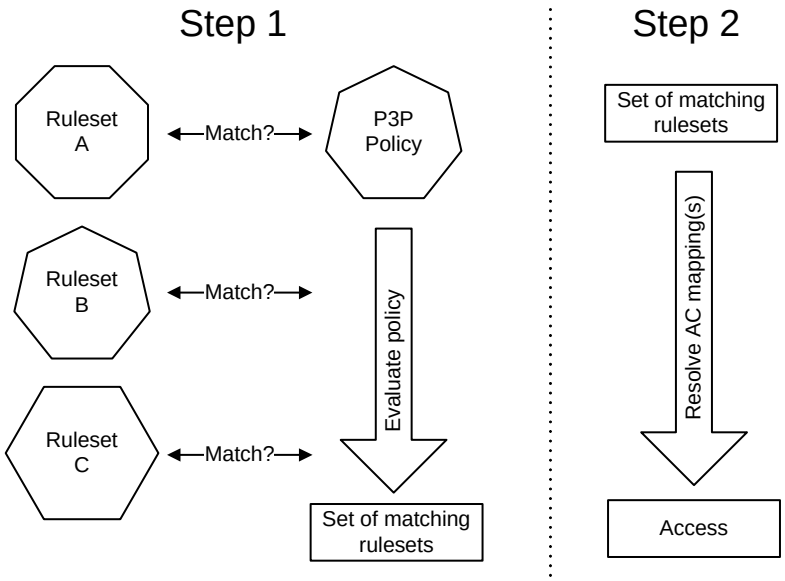


Figure 27. Resolving access with P3P

The advantage of the supplementary P3P mechanism is that privacy preferences need not be linked with a particular agent. They can also be expressed with respect to information usage. This enables access decisions to be based not only on identity but also on the intended usage of information. Thus it is possible to handle requests from previously unknown agents with improved accuracy. Instead of simply limiting unknown agents to public information it is possible to further evaluate their request. By comparing the agents intended usage of the information against the subject's privacy preferences additional privileges may be granted. The P3P based mechanism also formalises the process of disclosure (See above, section 3.2.4) by providing a common way of representing the agreement made between a subject and an agent.

Associated with the mechanism are also limitations and disadvantages. For the mechanism to allow unknown agents to be supported it has been assumed that the contract governing the usage of information, i.e. the P3P policy, is written by the agents alone. Consequently this limits the subject's ability to influence the usage of the information expressed in a P3P policy. The mechanism is left with only two options with respect to each ruleset. Either the usage of information is accepted or it is rejected. Another limitation of P3P in general, is that the specification does not provide any means of enforcing nor verifying an agent's compliance with their published policy. An important point raised by others as well [Grimm, Rossnagel 2000] [EPIC, Junkbusters 2000].

Furthermore, the specification states that "P3P declarations are positive, meaning that sites state what they do, rather than what they do not do" [World Wide Web Consortium 2002A s.1.1.3]. Thus potentially useful information is omitted. Finally P3P is not an easy technology to use. Policies as well as rulesets are difficult to define. A study has shown errors to be prominent [Cranor, Byers et al. 2003].

Despite its weaknesses P3P is a useful technology. It provides the means of representing the agreement made between a subject and an agent in a common format. Thus with P3P the intended usage of information can be described. This mechanism exploits this to allow access decisions to be made in situations where the agent is previously unknown. However, the inability to enforce and verify policies call for extra care when defining rulesets and mapping them onto access controls.

4.5.3. Anonymity and pseudonymity

Both anonymity and pseudonymity are states in which a subject can disclose information without revealing their true identity. This is an important quality that allows people to perform their daily tasks without constantly worrying about privacy.

Before we can provide anonymity and pseudonymity in ubiquitous computing environments it is necessary to define to what degree the identity of the subject needs to be protected. In the conceptual model of privacy an example is given in which a crowd allows a subject to be anonymous (See above, section 3.2.3). When claiming anonymity here it is assumed that the people in the crowd do not know the subject and that they do not actively attempt to identify the subject using, for example, event attendance details. Hence, anonymity does not require the identity of a subject to be perfectly hidden. It is enough if it is adequately obfuscated. This suggests that in a digital world, where a subject is connected to a larger network with a dynamic address, it could be enough to obfuscate the static address with which they are contacted. The same reasoning can be applied to pseudonymity, except that the subject needs to be able to retain a fabricated identity over time.

One method of achieving a basic level of anonymity/pseudonymity in the infrastructure would be by using dynamic addresses. For a subject this would involve the use of the context manager catalogue service (See above, section 4.4.2). The CMCS provides a context manager, i.e. subject, with a static address that maps to their dynamic context details. By using another or a second unknown static address the subject would in effect obfuscate their identity. For a requestor no other mechanism is required. This mechanism of course relies on a subject's context manager having a dynamic address. It also assumes that the subject does not reveal any other uniquely identifying information and that requestors use anonymous credentials. Furthermore, it also assumes an attacker does not go to any great length in trying to find out the true identity of the subject. For example, if the device reveals their host name this could be used to identify the subject.

Another method to achieve a more advanced level of anonymity/pseudonymity involves the breaking of the traceable path between a subject and requestor. For this to be achieved, techniques such as anonymity networks need to be employed. In the infrastructure the anonymising mechanism can be thought of as an external proxy component through which messages can be sent and received but not traced. How the component obfuscates the messages path is outside the scope of this work. From the point of view of the participants in the infrastructure the proxy will work as follows. When requests are to be sent anonymously, the requester contacts the proxy with the message and the address of the recipient. The component will then take responsibility for the delivery of the message and also for the return of the reply. As such there will be no traceable path between the requestor and the subject making the requestor anonymous. When requests are to be received anonymously, i.e. using a pseudonym, the proxy instead acts as a virtual recipient. The requestor sends the message to the proxy, in the belief that it is the subject. The proxy then forwards the request to the subject and also later returns the reply to the requestor. Hence, the message path becomes untraceable and the subject can remain known only by their pseudonym. It should be noted that even though this more advanced method provides a better level of protection it still relies on the

subject/requestor not revealing any uniquely identifying information themselves in the communication.

4.5.4. Notice

Notice is one of the core principles of privacy protection according to the fair information practise codes [Landesberg, Levin et al. 1998]. It stipulates that users should be made aware of when information is collected and how it will be used. Hence, notice enables informed decisions to be made of whether to utilise a service, allowing data to be collected, or not.

In this work notice is handled implicitly prior to disclosure. As described in the privacy model, the release of information is governed by an agreement that specifies the terms of the disclosure including any associated limitations or conditions (3.2.4.2). Therefore, if access has been granted it is assumed an agreement is in place and that the user has been properly notified by the other party.

The motivation for excluding explicit notification at the point of release can be found within the foundation of ubiquitous computing. The aim from the start has been to move computing into the background [Weiser, Gold et al.1999]. The use of notification, which draws attention to the system, is therefore believed to contradict a fundamental aspect of the work. Furthermore, the vision of ubiquitous computing also speaks of a world where large numbers of interconnected computing devices are embedded seamlessly and invisibly in our everyday environment. In such an environment it is not deemed feasible to expect users to respond well to explicit notifications as these are likely to be presented too often. Indeed even related work, that follows the fair information practise codes more closely, minimises explicit notification at the point of release through the use of personal privacy proxies acting on behalf of the users in accordance with their privacy preferences [Langheinrich 2002].

The consequences of the chosen approach are twofold. Firstly, it requires notification to occur prior to the point of release. This allows the users to perform their everyday tasks undisturbed, with the system in the background. However, it also implies that privacy preferences must be expressed when less

information is available about the situation(s) in which it will be disclosed. Whilst this can result in less accurate privacy control, it is deemed that the adoption of a restrictive access policy can to some extent be used to compensate for this. Secondly, there is no feedback when information is accessed. Again this ensures that users are not disturbed, which is good. The drawback is that it makes it difficult for users to monitor the disclosures that occur. This in turn negatively affects the ability to verify that agreements made are followed. But then again, even if the occurrence of disclosures could be monitored this would on the whole have little impact as the use of the information could still not be controlled.

4.6. Context communication format

An essential part of any context-aware infrastructure is the communication of contextual information. Simply letting sensors collect information is of little use. The gathered information needs to be passed on to services and/or applications to provide a user with an enhanced service. Whilst it is perhaps tempting in some situations to incorporate the sensing directly into the application, previous work has shown that it is preferable to keep the collection and use of information separate [Dey 2000B]. Thus independent of the situation, contextual information will always need to be communicated.

The communication of contextual information has traditionally been designed and implemented in an ad-hoc manner for each context-aware system developed. Hence there is no de facto standard to use. Consequently this has made collaborations between different systems unnecessarily complex, hindering the development of truly ubiquitous context-aware applications. This situation is of course not desirable.

To address the issue of communication both within the infrastructure and externally a uniform communication format has been developed together with a context vocabulary.

4.6.1. Objectives

The context communication format needs to meet three conditions: simplicity, universality, and versatility.

Simplicity - The format should be simple enough to allow it to be used with resource constrained devices. This is important since the majority of the agents deployed in an infrastructure are expected to run on resource constrained devices, after all it is they that yield the smallest form factor and the lowest cost. For example, both the sensors and pervasive interaction devices used generally have limited processing power, memory, and battery-life. Furthermore to fully meet the requirement of a decentralised infrastructure (See above, section 4.1.2.1) any agent must be able to operate independently within the infrastructure. Hence, at the minimum this implies that context consumers and producers can decode and encode contextual information in the communication format respectively. Finally it is also beneficial, if at all possible, for the format to be simple enough to allow the data to be humanly readable. This will aid developers to debug and test implemented agents.

Universality - The format should be universal, i.e. device and platform independent. This is necessary to support the diversity found in ubiquitous computing systems. The capabilities needed by a sensor for instance, are much different to those of an end user application. Furthermore, factors such as context, user preferences, costs, etc. also influence the choice of device and platform. Indeed previous and related work show that a large number of different devices have and are being used (See above, section 2.6.3). Thus it is essential that the context communication format does not impose limitations restricting its use to a particular device or platform.

Versatility - The format should be versatile. Context has been defined in terms of the relationship that exists between an entity and a piece of information (See above, section 3.1.1). Thus no restriction has been placed on what can be classified as context information, allowing it to vary with situations. It is therefore not possible to determine in advance exactly what information the format needs to support. A piece of context may for example be a name represented as a string, a location represented by coordinates, or even an image showing a user's current view. The context communication format must therefore be flexible enough to allow any data to be included.

4.6.2. Composite Capability / Preference Profiles

An existing format that largely meets the conditions is the Composite Capability/Preference Profiles (CC/PP). CC/PP is a standard that has been designed to allow device capabilities and users preferences to be communicated to remote services, allowing services to be tailored with respect to users and their devices. The CC/PP specification [World Wide Web Consortium 2004E] mainly focuses on conveying static device capabilities such as screen size, supported html version, etc. However, the format is not limited to this. It can accommodate a more general use through the definition of vocabulary extensions. Whilst CC/PP may not be suitable for modelling context [Indulska, Robinson et al. 2003], experiments have found it is sufficiently flexible to be used successfully in the communication of contextual information [Osbakk, Ryan 2002].

The RDF/XML format used to represent CC/PP profiles allows both the simplicity and universality objectives to be fulfilled. RDF is a structured XML format that can be encoded using plain text. Thus, in theory profiles can be inspected and manipulated by any device and platform capable of handling plain text. Naturally the device and platform support will be slightly more limited, in practise, since the profiles need to be interpreted as well. However, several minimal XML parsers exist [Wilson 2001A] [Wilson 2001B] [Aadland, Angel et al. 2002], one with a footprint as small as 6 kb [Scheemaecker 2003]. Also small higher-level RDF parsers exist [Megginson 2001]. Hence, CC/PP profiles are simple enough to be processed on resource constrained consumer devices, e.g. PDAs and mobile phones, as well as on embedded systems, e.g. on TINIs. CC/PP profiles are also universal as their representation makes them both device and platform independent.

The versatility objective however, is not fully met by the CC/PP format. Whilst the possibility to use vocabulary extension provides some degree of versatility, it is not enough by itself. Two important limitations remain. First of all a profile's hierarchy is only two-levels deep. Secondly the specification only defines basic data types. It is however possible to work around these issues, as

will be demonstrated in the following section (See above, section 4.6.3). Thus, for now it will be assumed that CC/PP fulfils the versatility objective.

Given that the CC/PP format can fulfil the objectives of a general context communication format, it was decided that this format was to be used. Thus, instead of defining a format specific to context communication, a vocabulary extension to CC/PP was developed. Profiles conveying contextual information will be referred to as 'context-profiles'.

4.6.3. Addressing CC/PP limitations

It has been stated that the CC/PP format's versatility is limited with respect to its hierarchical depth and the data types defined. This section will further describe these limitations and why they constitute an issue when CC/PP profiles are used for context communication. The section will also demonstrate how the limitations can be resolved. Please note that the techniques described should only be applied to information using the context vocabulary, defined later. Doing so will ensure that the resulting profile is backward compatible with ordinary CC/PP profiles.

4.6.3.1. Depth

A CC/PP profile's hierarchy is only two-levels deep. This is caused by the data structure of a profile. The CC/PP format uses two types of elements to structure data: components and attributes. Components are used to categorise groups of data, e.g. HardwarePlatform, whereas attributes identify and hold the data, e.g. ScreenSize. Thus the attained result is a simple two-level tree structure, illustrated in Figure 28.

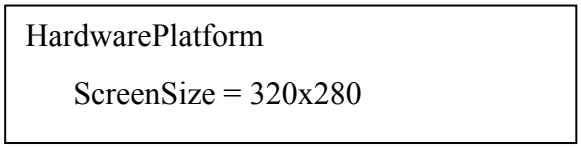


Figure 28. Two-level data structure

This two-level structure, although sufficient to communicate static device capabilities and simple user preferences, is not adequate to support the communication of general context information since essential metadata cannot be communicated. For example, it may be desirable to add a timestamp,

lifetime, or measure of accuracy to the data. Furthermore, complex pieces of context information may also require the data format to be described along with the data.

Different approaches can be taken to overcome this limitation and enable metadata to be present in a CC/PP profile. For instance, the functionality of RDF, which supports an arbitrary depth, can be utilised. Another alternative is to define custom complex CC/PP attributes. However, to preserve the structure of CC/PP, and hence its simplicity and universality, a third approach has been taken. The approach involves the use of a flattened three-level tree structure. To explain the chosen structure, the changes made when moving from a two-level structure to a flattened three-level structure will now be described.

Starting with the simple two-level CC/PP profile shown in Figure 28. To this structure a third level is added, dedicated to holding data and metadata. This results in the second level being used only as a data identifier. To allow basic data to be represented a common data holder must be defined. This data holder must be globally known and reserved for this use only. In this work ‘value’ has been reserved. The result is a three-level structure, illustrated in Figure 29.

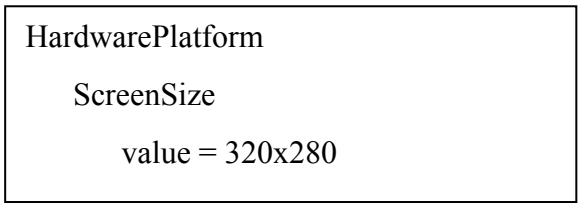


Figure 29. Three-level data structure

Now to achieve a flattened three-level structure, the category information and the data identifier are joined and represented by a component. The category information and data identifier are separated by punctuation, effectively creating a single string. Multiple separations, i.e. punctuations, can be used to create an arbitrary depth. Attributes are then used to represent the data holders. Thus a three-level structure can be represented using only two-levels. The result is referred to as flattened three-level structure and is illustrated in Figure 30.

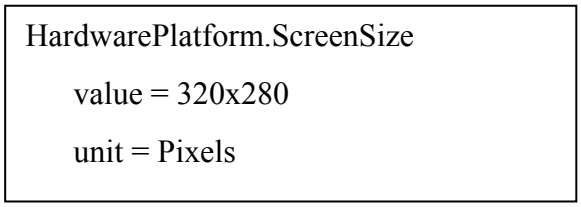


Figure 30. Flattened three-level data structure

With this flattened three-level structure context information and metadata can be added to a CC/PP profile.

4.6.3.2. Types of data

The CC/PP format defines a number of basic data types for the representation of attribute data [World Wide Web Consortium 2004E s.4.1]. Supported are the following simple data types: string, integer number, and rational number. In addition to these, two complex data types are supported: set of values and sequence of values.

Whilst the supported data types may be sufficient for the communication of static device capabilities and user preferences, it is not adequate for the communication of context information. Contextual information can take many forms including binary. Even in the simple case of adding textual context information care has to be taken to ensure none of the characters reserved for the xml/rdf structure are used. For example a P3P policy cannot be directly added to CC/PP policy as value since it uses xml. Another example is the current view of the user in the form of an image.

To address this issue a standardised way of encoding unsupported data needs to be specified. In accordance with the recommendation in the CC/PP specification, the use of complex data models has been avoided [World Wide Web Consortium 2004E s.2.1.2]. Instead it has been decided to utilise the mechanisms developed for the specification of Internet message bodies – RFC 1341 [Borenstein, Freed 1992]. Any unsupported data is encoded according to the “Base64 Content-Transfer-Encoding” [Borenstein, Freed 1992 s.5.2]. To indicate that that data has been encoded the “Content-Transfer-Encoding Header Field” [Borenstein, Freed 1992 s.5] is added as metadata. Furthermore, to aid the recipient in interpreting encoded data, it is recommended that the MIME type of encoded data also is added as metadata in a “Content-Type Header

Field" [Borenstein, Freed 1992 s.4]. Figure 31 illustrates the addition of unsupported data.

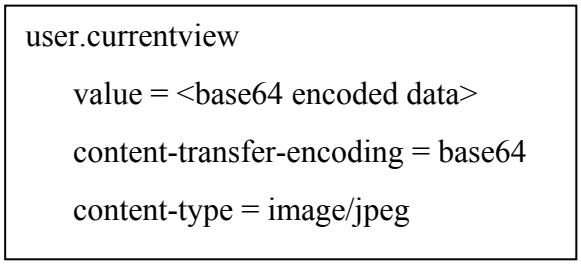


Figure 31. Adding unsupported data types using base64 encoding.

Using this technique it is possible to add otherwise unsupported data types to a profile.

4.6.4. Context vocabulary

The developed vocabulary has been inspired by the Platform for Privacy Preferences (P3P) [World Wide Web Consortium 2002A] and whenever possible its names and structures have been used. There are several reasons for doing this. Firstly, the notation used by the P3P is both standardised and well known. Secondly, it covers some of the basic context information that is desirable to include in the vocabulary. Finally, a P3P based privacy protection mechanism has been developed in conjunction with the context communication format and by using the same names and structures in the context vocabulary the administration of this mechanism is simplified.

The vocabulary consists of three parts: categories, data identifiers, and data holders.

Beginning with the categories, their aim is to allow related data to be organised into groups. This makes the data easier to find and allows frequent data identifiers to be used more than once in the vocabulary. There are four broad categories: 'user', 'business', 'activity', and 'location'. The categories have been identified from the characteristics often considered by context-aware applications, the "who's, where's, when's, and what's" of entities [Dey 2000B p.4]. The 'user' and 'business' categories correspond to the 'who's'. They group data related to an individual or an organisation (or non-human entity) respectively. The 'activity' category corresponds to the 'what's' and includes

information about an entity's current activity. Finally the 'location' category corresponds to the 'where's'. It groups information representing an entity's physical and possibly virtual location. Please note that there is no category for temporal information, i.e. the 'when's'. This information is considered to be metadata and will be dealt with later.

Under each of the categories a number of data identifiers have been defined. The user category contains information structures like 'name', 'bdate', and 'home-info'. These hold the individual's name, date of birth, and home contact details respectively. Similarly the business category contains structures like 'name' and 'contact-info' to hold the name of the entity and their contact details. In the activity category, a 'type' and a 'note' identifier can be found. They hold the type of the activity and a short description respectively. Two structures 'start' and 'end' also exist, allowing the time frame of an activity to be described. Finally under the location category a 'coordinates' and a 'place' identifier can be found. They hold position information in the form of a longitude-latitude pair and a place name respectively. Furthermore a 'contact' structure also exists under the locations category. This structure holds any location specific contact information.

Together the categories and the data identifiers form the components that identify particular pieces of context in a context profile. Please note that the structures have deeper hierarchies. For example 'user.name' is a higher level representation of 'user.name.family' etc. Figure 32 lists the resulting components in the vocabulary. Further details can be found in Appendix A.

Category	Data Identifier	⇒ Component
user	name†	user.name†
user	bdate†	user.bdate†
user	home-info†	user.home-info†
business	name†	business.name†
business	contact-info†	business.contact-info†
activity	type	activity.type
activity	note	activity.note
activity	start†	activity.start†
activity	end†	activity.end†
location	coordinates	location.coordinates
location	place	location.place
location	contact†	location.contact†

† Structure

Figure 32. Context vocabulary components.

Finally the vocabulary defines several data holders, see Figure 33. These form the attributes in the context profile. The most fundamental data holder defined in the vocabulary is 'value'. Its purpose is to hold the actual context information in the flattened three-level structure (See above, section 4.6.3.1). The remaining data holders defined in the vocabulary only contain metadata. They are: 'content-transfer-encoding', 'content-type', 'timestamp', 'accuracy', 'confidence', 'lifetime'. The 'content-transfer-encoding' and the 'content-type' hold metadata that describe the format and content of the data contained in the 'value' attribute. They are used when adding data that cannot be supported by the built in data types (See above, section 4.6.3.2). The 'timestamp' attribute allows the data to be accompanied with a date and time, generally showing when the information was captured. Thus, the 'timestamp' takes into account the 'when's'. The 'accuracy' and 'confidence' attributes allow estimates of the quality to be included. Finally the 'lifetime' attribute can be used to represent the expected validity period of the data. Further details can be found in Appendix A.

Data holder ⇒ Attribute
value
content-type
content-transfer-encoding
timestamp
accuracy
confidence
lifetime

Figure 33. Context vocabulary attributes

With the described components and attributes the context vocabulary enables basic contextual information to be communicated. Naturally the support is limited to only a small subset of all possible contexts that it may be desirable to communicate. It has never been the aim to provide a complete vocabulary. Instead the vocabulary has been developed for the needs of this infrastructure. It is then expected that further compatible vocabulary extensions will be defined as required.

4.7. Summary

In this chapter the design of a privacy enhancing infrastructure has been described.

The requirements capture has involved extracting potential requirements from the aim, the background, and the employed models then reviewing these to form an effective requirements set. The set consists of three categories of requirements privacy, functional, and miscellaneous. The privacy category is the most important with requirements such as: retain offline level of privacy, customisable balance of privacy, ability to handle both known and unknown agents, decentralised structure, and security being covered.

Also presented is the strategy taken during the development as well as the scope of the infrastructure. The strategy has involved prioritising privacy throughout the work rather than functionality. It has also been an objective to produce a modular design that allows customisation. The scope of the infrastructure has been adjusted to fit area of ubiquitous computing. To reduce the required

resources the depth of the context model has been restricted. Also to simplify the development a specific type of device, namely, personal digital assistants has been targeted.

To meet the requirements, a decentralised component based architecture has been designed for the infrastructure. The main component in the architecture is the context manager. Each entity in the infrastructure is associated with at least one context manager that handles their contextual information. Interacting with the context managers are agents, context consumers and context producers, that retrieve or store contextual information, respectively. There is also a data storage component and a catalogue service component in the architecture, providing storage of date and address resolution, respectively.

For privacy protection the focus is on access control. Thus, three different access control mechanisms have been presented. A classification and clearance scheme that uses levels to express information sensitivity and the trustworthiness of context consumers. A role based access control mechanism that uses roles and permission to specify access. And a mechanism based on the platform for privacy preferences that enables preferences to be expressed with respect to privacy polices. Discussed is also anonymity/pseudonymity and briefly authentication.

A context communication format has also been presented. In developing the format the properties of simplicity, universality, and versatility has been emphasised. Subsequently, the Composite Capability Preference Profiles, an existing data format, has been extended for more general use.

CHAPTER 5

IMPLEMENTATION

The previous chapter has described the architecture of a privacy-enhancing infrastructure. It has also described some alternatives for privacy protection and a context communication format. Together these partially fulfil the requirements. Before the infrastructure can be evaluated however, it needs to be implemented. The implementation also provides the remaining features necessary to meet the requirements.

This chapter presents the implementation of the infrastructure developed. Particular attention has been given to explaining implementation specific details, relevant to the fulfilment of requirements. The chapter also presents a number of agents developed to test and demonstrate the infrastructure.

5.1. Overview

Following the previously presented architecture (See above, section 4.4), the implementation consists of several components. Each component will be described in detail in this chapter and an overview is provided by Figure 34.

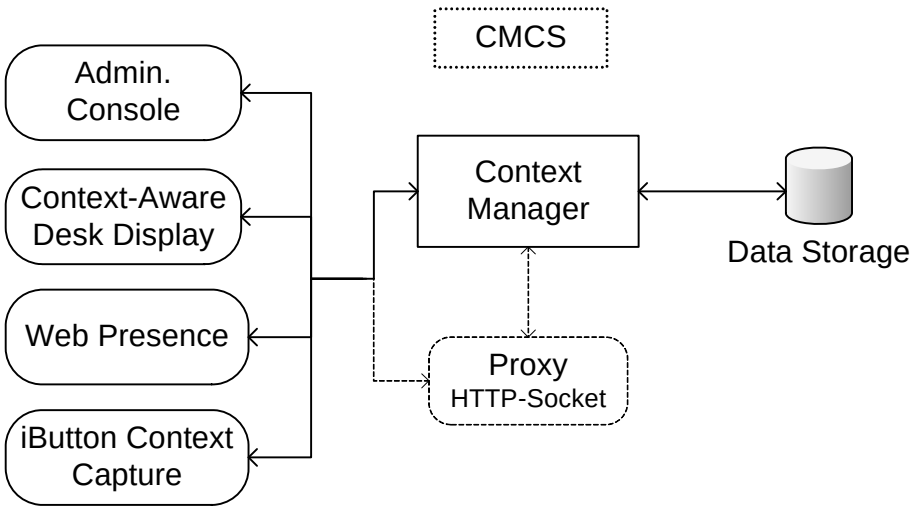


Figure 34. Overview of implemented infrastructure components.

5.2. Platform

All the components in the infrastructure have been implemented in Java. This makes the infrastructure platform independent. Since the infrastructure is targeted at resource constrained device types, personal digital assistants (See above, section 4.3.2), it has been developed for the Micro Edition of the Java 2 Platform (J2ME) [Sun Microsystems 2002]. More precisely the infrastructure has been developed for the Connected Device Configuration [Java Community Process 2002A], Foundation Profile [Java Community Process 2002B], and the Personal Profile [Java Community Process 2002C]. Together these components form a complete Java runtime environment that is suitable for use on resource constrained devices. It will henceforth be referred to as J2ME CDC/PP.

A distinct advantage of the J2ME CDC/PP platform is its compatibility with both Java 2 Platform, Standard Edition (J2SE) and Personal Java Application Environment (PJAE) [Sun Microsystems 1998]. This enables the infrastructure to be deployed on a wide range of devices. The compatibility with J2SE platform allows the application to run on more powerful traditional computing devices, e.g. desktops and servers, whilst PJAE provides backward compatibility with older pre-J2ME environments, e.g. PDAs and Smartphones.

5.3. Context Manager

As previously stated the Context Managers are the central components in the infrastructure (See above, section 4.4.1). It is with them the other infrastructure components interact to store and receive contextual information. This interaction is performed using a request/response protocol, where the context manager takes the server role. Hence the context managers are to run as background processes in the infrastructure, constantly ready to receive requests from agents. The requests form the inward flow of information to the context manager. Each request is processed by the context manager upon being received. Once processed the context manager returns an appropriate response. The responses form the outward flow of information. Figure 35 illustrates the inward/outward flow of information.

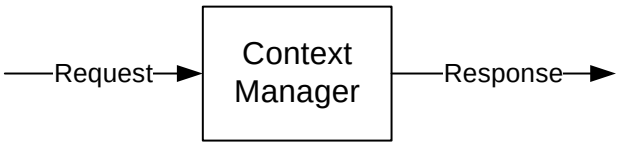


Figure 35. Inward and outward flow of information

The context manager's handling of incoming requests can be broken down into four stages, as illustrated in Figure 36, each addressing a different aspect. The first stage is inward communication. During this stage the context manager receives the incoming request and, if required, decrypts the request. The second stage is privacy protection. It is during this stage the subject's privacy preferences are enforced. This includes verifying the requestor's supplied credentials, evaluating their access, and filtering the request to remove any unauthorised actions. The third stage is request fulfilment. During this stage the request is executed. This involves going through a pre-processing procedure, performing the authorised actions, and carrying out a post-processing procedure. The fourth and last stage is outward communication. At this stage the response is encrypted, if required, and the response is returned to the requestor.

Stage 1: Communication – inward

- Receive request
- Decrypt request

Stage 2: Privacy protection

- Authenticate
- Evaluate access
- Filter request

Stage 3: Request fulfilment

- Pre-processing
- Perform actions
- Post-processing

Stage 4: Communication – outward

- Encrypt response
- Return response

Figure 36. The CM's request handling stages and their subsidiary steps.

The implementation of these four stages will now be described.

5.3.1. Communication

Several of the captured requirements concern the communication in the infrastructure either directly or indirectly (See above, section 4.1). It is therefore not surprising that communication plays an important part in the handling of requests. In fact given the architecture, communication is essential even for the most basic operations such as allowing agents to interact with the context managers to store and retrieve information (See above, section 4.4.3).

To meet the captured communication requirements a customisable communication scheme has been constructed. The scheme focuses on three different aspects of communication namely the method of communication, the transport mechanism, and confidentiality. This is illustrated in Figure 37.

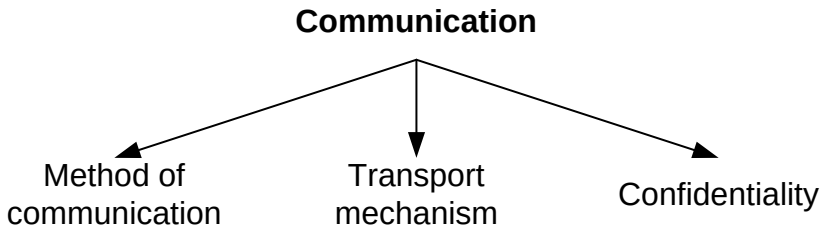


Figure 37. The three aspects of communication.

5.3.1.1. Method of communication

A method of communication has been implemented that suites the request/response style interaction employed by the context manager.

The method splits interaction dialogues into sequences of messages, where each message is either a request or a response. A message consists of two parts: a header and a body. The header is primarily used to communicate information that applies to the message as a whole. This includes information such as the status of the request/response, authentication details, etc. It can also be used to issue special commands, to retrieve a public key for example. The body on the other hand contains the action to be performed. An action can for example be to store or retrieve a piece of contextual information. Within a single request multiple actions can be performed.

In the infrastructure the messages are represented using an extension to the context communication format described earlier (See above, section 4.5.4). The extension allows implementation specific information to be encoded.

Specifically it provides support for the header-body structure and extends the vocabulary to handle various actions. It should be noted that any messages represented are still valid 'context profiles' and compatible with the CC/PP specification [World Wide Web Consortium 2004E]. The extension merely provides the semantics necessary to create and interpret infrastructure specific messages accurately.

Context profile extension

Support for the header-body structure is added to the 'context profiles' by introducing a component, reserved for header information. The header component, just as any other component, then uses attributes to hold data. The rest of the context profile forms the body. This approach ensures that the context-profile is backward compatible, as only additions to the vocabulary are made. However it does limit the type of information that can be embedded in the header. For example, it is not possible for a piece of information stored in the header to have associated metadata. This restriction is acceptable though, as the header must only contain metadata with respect to the whole message or special commands.

The header component is mandatory when communicating with a context manager. Furthermore, every request must at least contain a 'username' attribute in the header, where an anonymous use is supported with the data value 'anonymous'. In addition to the 'username' attribute, there are several optional attributes including 'password', 'encrypteddata', 'publickey', 'status', etc. Figure 38 provides additional information about the frequently used header attributes. For a more comprehensive listing consult Appendix B.

Header attribute	**Description**
username	The requester's identity.
password	Password authenticating the claimed identity.
encrypted	The name of the employed cipher.
encryptedData	Holds encrypted data.
command	Any command to be performed, e.g. synchronise.
status	The request's status.

Figure 38. Frequently used header attributes

Additional attributes are also defined for use within the message body. The message body is the optional part of the message that carries contextual information. The information is embedded directly in the 'context profiles' as previously described using components and attributes (See above, section 4.6.4). The additional attributes defined are specific for the context manager implementation and are applied individually on each component in the context profile. Several attributes are included in the extension including 'action' and 'status'. The 'action' attribute, for example, is mandatory if the context manager is to process the component. It can hold the values 'read', 'write', and 'history'. They correspond to the retrieval of current contextual information, the storing of contextual information, and the retrieval of past contextual information. Figure 39 provides additional information about the frequently used body attributes. For a more comprehensive listing consult Appendix B.

Body attribute	**Description**
action	The action to be performed.
status	The status of the individual action.

Figure 39. Frequently used body attributes

5.3.1.2. Transport mechanism

The context manager has been implemented to use plug-ins to transport messages. This allows the low-level details of the transport mechanism to be transparent to the context manager. Hence, the context manager is only responsible for the handling of transport plug-ins. The details of the transport mechanism are then implemented in the plug-ins.

There are two types of plug-ins being used, server plug-ins and client plug-ins. The server plug-ins provide the ability to receive requests and return responses. The client plug-ins on the other hand allow requests to be made and responses to be received. Together a plug-in pair provides the necessary mechanism to support a two-way transportation of messages, see Figure 40.

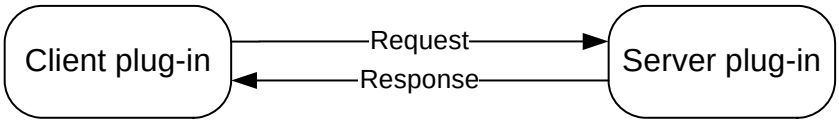


Figure 40. Client and server plug-ins.

Each client and server pair is implemented to handle a particular mode of transport. Furthermore the client-server pairs are exchangeable and multiple pairs can co-exist. This ensures that the mode of transport is fully customised, as required (See above, section 4.1.2.2).

Transport plug-ins

Whilst the details of the transport mechanism differ from plug-in to plug-in, their interface towards the infrastructure components must follow strict guidelines.

First of all, the plug-ins must follow a specified naming scheme and be located in a known package. The naming scheme divides the name of the class into two parts a plug-in name and a suffix. The plug-in name can be any combination of lowercase letters but must be the same for both plug-ins in a pair. The suffix on the other hand varies with the type of plug-in, where the client plug-in must use the suffix 'TCP' and the server plug-in 'TSP'. Note that the suffix must always be in capital letters. The package the plug-in classes must be located in is 'net.osbakk.pi.cm.plugins'. It is important that the naming scheme is followed and that the classes are placed in the right package as the context manager will use this information when dynamically loading plug-in classes.

Secondly, each plug-in must implement an interface that corresponds to its type. Starting with the client plug-in, the less complex of the pair, it must implement the 'TCPInterface', outlined in Figure 41. The TCPInterface mandates that methods to initialise the plug-in and to post requests are implemented. Then there is the server plug-in. It must implement the 'TSPInterface', outlined in Figure 42. The TSPInterface requires seven methods to be implemented including methods to initialise the plug-in, to control and retrieve its state, to retrieve any associated addresses, to post responses, and to attach communication event listeners. The communication event listeners provide the link between the plug-in and context manager and must be used to pass on incoming messages as events. Full details of both interfaces can be found in Appendix C.2 and C.3 respectively. The details of the communication event listener and the events that must be sent by the transport server plug-in are available in Appendix C.4 and C.5 respectively.

Name: <plug-in name>TCP†
Package: net.osbakk.pi.cm.plugins
Impl. interfaces: TCPInterface
Methods:

init	post

Notes: N/A
†<plug-in name> follows regular expression [a-z]+

Figure 41. Transport client plug-in specification.

Name: <plug-in name>TSP†
Package: net.osbakk.pi.cm.plugins
Impl. interfaces: TSPInterface
Methods:

init	isConnected	addComEventListener
connect	getAddresses	
disconnect	reply	

Notes: N/A
†<plug-in name> follows regular expression [a-z]+

Figure 42. Transport server plug-in specification.

Socket Plug-in implementation

A transport plug-in pair has been implemented that uses TCP/IP sockets as the means of transport. The socket plug-in pair transports the messages as a single stream of characters only preceded by the length of the message being sent. The length of the message consists of an integer ending with a hash character. The implementation works as follows and its lifecycle is illustrated in Figure 43.

Beginning with the server side. When the context manager is started it checks the configuration to see which plug-ins to load. Assuming the context manager has been configured to use the socket server plug-in, the plug-in is loaded. Once loaded the plug-in is initialised. During the initialisation the plug-in parses the properties file and extracts the address of the server. If not manually configured, this information is detected automatically. When this is done the context manager activates the plug-in by calling the connect method, prompting it to start listening for connection on the default port. If for some reason the port is unavailable, another port will be chosen automatically and the addresses

updated appropriately. Once listening to the port the plug-in sits in the connected state, ready to receive requests from agents and responses from the context manager. Upon receiving a request it will wrap the message, together with a connection identifier, in an event and pass it on to the context manager using the communication event listener. The request will then be processed by the context manager before a response is returned. Upon receiving a reply event from the context manager the server plug-in will unwrap the event and extract the message and the connection identifier. The message containing the response will then be returned to the requestor using the appropriate connection. Once done the server plug-in returns to the connected state. The plug-in is capable of processing multiple request/responses at once. Finally when the context manager is shutting down it will deactivate the server plug-in. This causes the plug-in to stop listening for new connections and terminate with the context manager.

On the client side, the process starts when the first message is going to be sent to a particular context manager using the socket protocol. The agent first loads the client plug-in. The plug-in is then initialised with the context manager's address. Once initialised the plug-in sits in a disconnected state ready to post requests. Then when a message is to be posted a connection is opened to the specified address and a request sent. The client plug-in will then be placed in a connected state, waiting for a response. When the client plug-in receives the response, it closes the connection and returns the received message to the agent. If no timely response is received the plug-in will time-out. This will also close the connection, but instead of returning a response an exception will be thrown. Once the connection is closed the plug-in will return to the disconnected state. The plug-in then remains in the disconnected state until another request is posted or the agent terminates it.

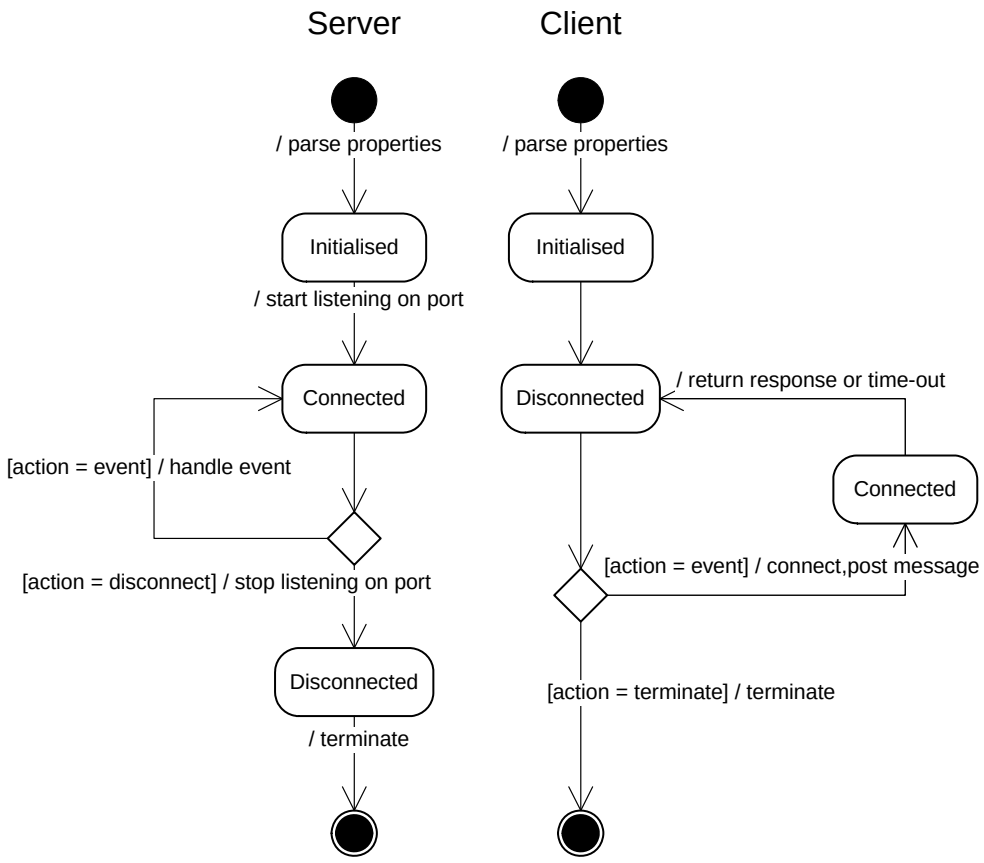


Figure 43. Socket server and client plug-in lifecycles.

5.3.1.3. Confidentiality

To ensure the confidentiality of the communication another plug-in based mechanism has been implemented. The mechanism uses cryptographic plug-ins to secure the interaction dialogue. Each cryptographic plug-in developed is responsible for both the encryption and decryption of messages using their own particular function(s). Furthermore, the plug-in must be able to generate suitable keys. Consequently each plug-in provides a complete cryptographic solution including encryption, decryption, and key generation.

The implemented mechanism does not impose any direct limitation on what type of cryptographic functions to be used, though it has been designed with public key cryptography in mind. This allows differences both in requirements as well as legislation to be supported. Moreover just as the transport plug-ins are exchangeable so are the cryptographic plug-ins. Hence a selection of cryptographic functionality can be provided using different plug-ins. The mechanism also allows the different cryptographic plug-ins to co-exist, ensuring wide runtime support.

Cryptographic plug-ins

All cryptographic plug-ins must have a common interface towards the context manager. This interface is defined by the following criteria.

First of all, the cryptographic plug-ins must follow a particular naming scheme. The scheme is similar to the one used by transport plug-ins. Hence the name of the class consists of a plug-in name and a suffix. The plug-in name can be any combination of lower case letters, whilst the suffix must be 'EDP' in capital letters. The plug-in must also be located in the plug-in package 'net.osbakk.pi.cm.plugins'. Once again these rules are important as this information is used when dynamically loading plug-in classes.

Secondly, all cryptographic plug-ins must implement the 'EDPInterface', outlined in Figure 44. The interface requires four methods to be implemented including methods to initialise the plug-in, to generate keys, to encrypt data, and to decrypt data. With these methods the context manager can interact with the plug-in to encrypt and decrypt information. To provide a uniform support for various key types, the interface requires individual keys to be formatted as byte arrays. For key pairs a wrapper class is used. Full details of the 'EDPInterface' and the key pair class can be found in Appendix C.6 and C.7 respectively.

Name: <plug-in name>EDP†
Package: net.osbakk.pi.cm.plugins
Impl. interfaces: EDPInterface
Methods:

init	encrypt	decrypt
generateKeys		

Notes: N/A
†<plug-in name> follows regular expression [a-z]+

Figure 44. Cryptographic plug-in specification.

RSA AES Plug-in

A cryptographic plug-in has been developed that employs both an asymmetric cipher, RSA [Rivest, Shamir et al. 1977], and a symmetric cipher, AES [NIST 2001]. The RSAAES plug-in, as it will be referred to, aims to strike a balance between security, manageable key distribution, and performance. To implement

the plug-in the Bouncy Castle crypto package [Bouncy Castle 2003] has been used.

The implementation works as follows. Upon either receiving or sending the first RSAAES encrypted message the context manager will create an instance of the plug-in. This instance will then also be used for subsequent request/responses. Once created the RSAAES plug-in is initialised. The initialisation sets up the BouncyCastle cryptographic provider and a pseudo-random number generator. When done the plug-in is ready to generate keys, encrypt data, and decrypt data using the BouncyCastle API [Bouncy Castle 2003]. The key generation process is straightforward. It simply creates a new pseudo-random RSA key pair and encodes the generated public and private keys as byte arrays. In a typical scenario this process is only performed once. Only if the private key is compromised, does a new RSA key pair need to be generated.

The encryption process is somewhat more complex and can be divided into four steps. The first step generates a new pseudo-random secret key for use with the AES cipher. The second step encrypts the generated secret key using the RSA cipher and the message recipient's public key. Once this is done the encrypted secret key is added to the output stream together with its length encoded using two bytes. The third step generates a 16 byte pseudo-random initialisation vector for use with the AES cipher. This initialisation vector is also added to the output stream. Finally, the fourth step encrypts the sensitive data and adds it to the output stream. The encryption is performed using the AES cipher, the generated secret key, and the initialisation vector. The resulting output stream is illustrated in Figure 45.

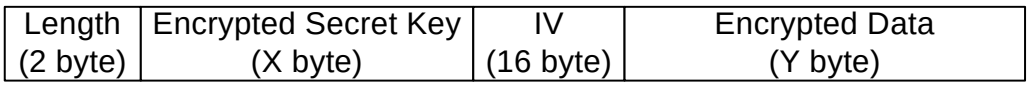


Figure 45. Data format of RSA-AES plug-in.

The decryption process is similar but retrieves information from the input stream rather than using the pseudo-random number generator. The process consists of three steps. The first retrieves and decrypts the secret key embedded in the input stream. The decryption is performed using the RSA cipher and the recipient's private key. The second step retrieves the initialisation vector from

the input stream. No processing is required. Finally, the third and last step decrypts the actual data. For this the AES cipher, together with the decrypted secret key and the embedded initialisation vector, is used.

5.3.2. Privacy protection

The second stage, privacy protection, handles the protection of a subject's privacy. During the design of the infrastructure a number of different mechanisms and approaches has been evaluated (See above, section 4.5). This section will describe which of these mechanisms has been chosen to be implemented by the developed infrastructure and explain the choice.

The context manager implements two aspects of privacy protection previously described, namely authentication and access control. This is illustrated in Figure 46. Although the authentication mechanism described here will take the issue of anonymity into consideration, this last aspect will utilise an external mechanism described later (See above, section 5.5.2.2).

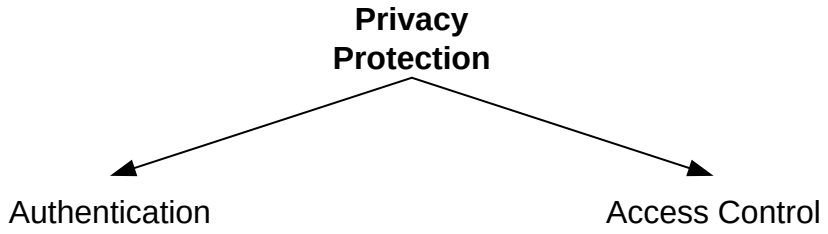


Figure 46. Privacy protection aspects.

5.3.2.1. Authentication

Of the two alternative authentication mechanisms previously short listed (See above, section 4.5.1) the password-based mechanism was chosen to be implemented.

This choice is motivated by the simplicity of the mechanism and the wide device support it yields. By using the less resource demanding password based mechanism the performance of the infrastructure is also expected to be better. Furthermore, in a research environment the probability of a subject being exposed to replay attacks is considered to be low. Hence, the increased risk associated with the use of the weaker mechanism is judged to be offset by its benefits. It should also be noted that it would be possible at a later stage to compliment the infrastructure with support for a challenge-response mechanism.

Since the infrastructure has been implemented to be stateless, the authentication is performed on a per message basis. This means that every message sent to a subject's context manager must contain an identity. The message must also contain a password that verifies authenticity of the claim unless the requestor asks to be anonymous. As already described the username and password are added to the header of a message using the attributes 'username' and 'password' respectively (See above, section 5.3.1.1). Supplying the username 'anonymous' marks the request as coming from an anonymous agent.

Upon receiving a non-anonymous request the context manager will authenticate the requestor by comparing the supplied password with that stored by the context manager for the identity in question. The identity could of course be a pseudonym. If there is a match then the requestor's claim has been positively verified and the request can be further processed. If on the other hand the password does not match, the requestor is assumed to be a rogue user and the request will not be further processed. Anonymous requests, if allowed by the subject, will always be positively verified. They will incur a slight delay in their processing though, to discourage unnecessary use.

5.3.2.2. Access control

To control access to a subject's context both the role based mechanism (See above, section 4.5.2.2) and the P3P based mechanism (See above, section 4.5.2.3) were chosen to be used. This combination allows privacy preferences to be expressed with respect to be unknown and known users.

The motivation for choosing the role based mechanism over the classification and clearance scheme is its scalability. As previously demonstrated the probability that the CCS mechanism can accurately represent a subject's preferences diminishes as the number of agents and context items increase (See above, section 4.5.2.1). Even in deployment of limited size the number of agents and context items are deemed to be large enough to cause considerable inaccuracies. Hence, the higher initial effort required to set up associated with the RBAC is judged to be preferential.

In the context manager the RBAC mechanism has been implemented as a two step process, illustrated in Figure 47. The first step evaluates an agent's access. This involves calculating its overall permission by aggregating permission granted by all the subject's roles, including those associated with any matching P3P rulesets, and then superimposing their personal permission. As previously asserted roles are always aggregated such that the best possible access is granted (See above, section 4.5.2.2). The resulting permission represents the effective access granted to the agent for that particular request. To make the access control mechanism efficient an agent's overall permission is cached. Naturally, in the event that a subject's privacy preferences change this cache is cleared. The second step filters incoming requests. During this step any unauthorised entries are discarded. This isolates the privacy protection stage from the request fulfilment stage, as any request passed on should be fulfilled, if possible. If at a later stage it becomes desirable to alter or exchange the access control mechanism, this separation will be beneficial.

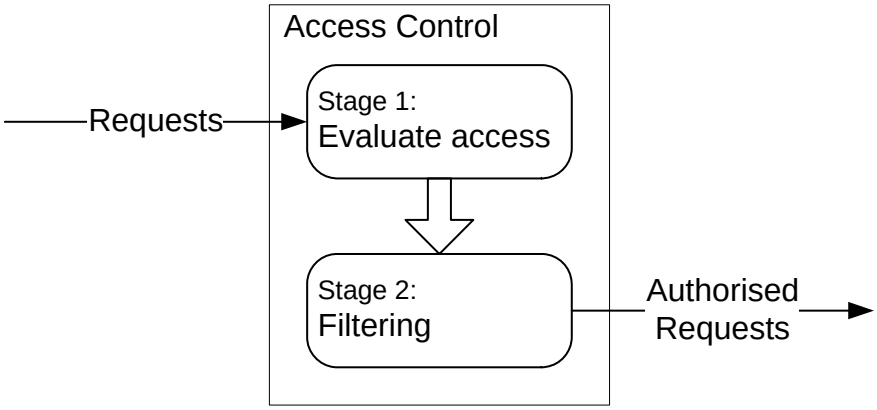


Figure 47. Access control steps

5.3.3. Request fulfilment

The third stage of the context manager's request handling concerns the fulfilment of requests. Within this stage there are three distinct parts: data storage, synchronisation, and extensions. This is illustrated in Figure 48.

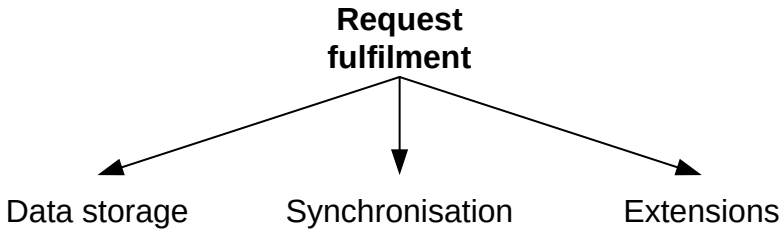


Figure 48. Request fulfilment aspects.

At the very basic level the request fulfilment stage enables contextual information to be stored and retrieved. This involves executing the read, write, and history actions embedded in the requests (See above, section 5.3.1.1) and interacting with the data storage. The request fulfilment stage also provides the necessary mechanism to support the synchronisation of information, as required (See above, section 4.1.2.2). Furthermore, through the implementation of extensions additional advanced functionality can be incorporated as well. For example, the use of extensions allows tasks such as context aggregation and evaluation to be performed. It also provides the mechanism by which the required support for context triggered events and request triggered context collection (See above, section 4.1.2.2) can be implemented.

5.3.3.1. Data storage

To store and retrieve information a context manager must use data storage. The data storage is a separate infrastructure component (See above, section 4.4.3) with which the context manager interacts. To ensure that no restrictions are placed on what type of data store that can be used (See above, section 4.1.2.2) a driver based approach has been taken when implementing the interaction. This involves using an exchangeable data storage driver to link the context manager and external data storage.

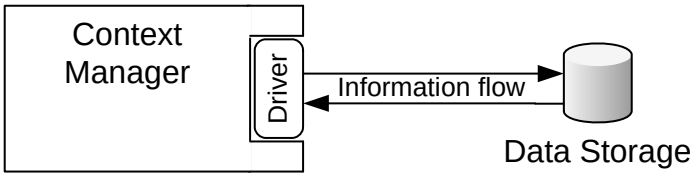


Figure 49. Data storage driver.

Hence the data storage driver provides the interface between the internal workings of the context manager and the external data storage, as illustrated in

Figure 49. As such the context manager enjoys a uniform mechanism for storing and retrieving information, fully independent of the type of external data storage being used making the interaction between the context manager and the data storage seamless, as required (See above, section 4.4.3). Thus, whilst the context manager is in charge of handling the storage of information, it is the data storage driver that executes the actual storage operation.

Data items

To minimise the complexity of communicating with the driver, the context manager organises information into data items.

A data item represents a particular piece of information and is identified using a category and a key. The categories separate different groups of information. For example one category is used for contextual data whilst others are used for data concerning access control etc. The key identifies an entry within a particular group, for example a particular context or user. Thus the category and key together provide a unique identifier.

Over time however, there may be several distinct occurrences of a single data item. Hence to identify the value of a particular piece of information a timestamp must be used together with the category and key. This will then allow the value of a data item, at a particular point in time, to be identified. The resulting data hierarchy is illustrated in Figure 50.

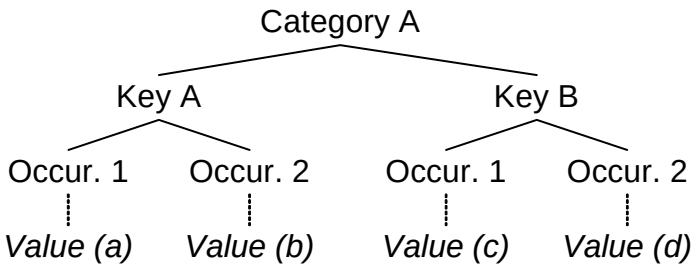


Figure 50. Data hierarchy.

The value of the data items takes the form of serializable Java objects. Hence, the usage of an external data storage does not appear different from an internal hashtable to the Context Manager. It also makes it easier for the data storage driver to handle the information and convert it, if necessary, to the format required by the data storage.

Finally it should be noted that data items constitute the format in which information is delivered to the driver from the context manager. It is not necessarily how the information is physically stored. This is determined by the data storage and its driver.

Datastorage driver

The data storage driver is required to fulfil a number of criteria.

First of all, the main driver class must be called ‘DataStorageDriver’ and have the package name ‘net.osbakk.pi.cm.dataStorage’. This ensures that the context manager can always find the driver. It also guarantees that only one driver can be loaded at any one time by the Java runtime environment. Both are essential aspects given that the driver is critical for the operation of the context manager.

Secondly, the main driver class must also implement the data storage interface, ‘DataStorageDriverInterface’, outlined in Figure 51. The interface stipulates the mandatory methods needed to initialise the driver, connect and disconnect from the data storage, set data, get data, delete data, etc. Full details of the interface can be found in the Appendix C.1.

Finally, the data storage driver is required to support historic information. Hence all information stored must contain a timestamp. The timestamp can either represent when the information was recorded or when it is stored, the context producers decide which. How long information is stored and to what extent this length is customisable is determined by the driver implementation. It is therefore optional for a data storage to retain historic information. The reason for this is that it allows the infrastructure to operate even when storage capacity is limited, albeit with limited functionality.

Name: DataStorageDriver

Package: net.osbakk.pi.cm.dataStorage

Impl. interfaces: DataStorageDriverInterface

Methods:

init	getData	getOccurences
connect	deleteData†	getLastOccurencesTime
disconnect	getCategories	clear
isConnected	getKeys	flush
setData	getKeysTable	

Notes: The driver must support historic information.

† 3 methods with different signatures.

Figure 51. Data storage driver specification.

Memory/File driver implementation

A memory/file driver has been implemented for the infrastructure. The driver uses internal memory to store data during runtime and a file to ensure persistency.

The implementation works as follows and its lifecycle is illustrated in Figure 52. When initialised the driver parses the properties to read in the configuration. The configurable entries include the history length, the location of the data file, and the filename. Then once the connect method is called the driver loads the configured data file into memory, making any stored information available to the context manager. Once loaded, the context manager can perform operations, flush data, and disconnect the data storage. The operations work on the data stored in memory and provide the means for setting data, getting data, deleting data, etc. The flush method writes the result of any performed operations to disk. Finally, the disconnect method makes sure the information in memory has been written and closes the data file. Consequently, this makes further operations impossible. Hence the disconnect method is only invoked when the context manager is being shutdown.

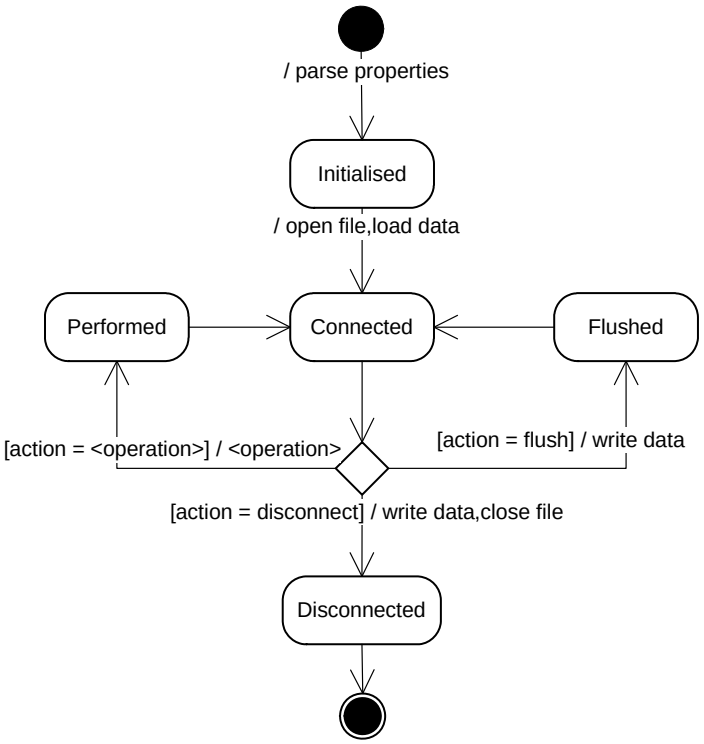


Figure 52. Memory/File driver state chart.

The memory/file driver implemented retains a user configurable amount of historic information. To minimise processing the history length is specified as the number of occurrences of data items to retain. The driver then uses a rollover stack in which the items are sorted with respect to their timestamps. Hence when the stack is full and a request is received from the context manager to store a new piece of information, the driver first removes the oldest item in the stack before adding the new. In this way the most recent information is retained.

5.3.3.2. Synchronisation

In addition to handling the retrieval and storage of information, the context manager has been implemented to support synchronisation, as required (See above, section 4.1.2.2). The implementation allows both contextual information and configurations, e.g. access control settings, to be synchronised between context managers.

Process

In the infrastructure synchronisation is always performed between two context managers at a time. The process is triggered when an agent sends a synchronisation request to a context manager. For example, this can occur as a

result of user interaction or be a predefined behaviour. The context manager receiving the request becomes the initiator of the synchronisation. It is the initiator that contacts the second context manager, referred to as the responder, and executes the synchronisation protocol.

Synchronisation requests

A synchronisation request is a request that contains the command 'synchronise' in the header of the context profile. This command prompts the context manager to perform a synchronisation.

Each synchronisation request is required to contain a 'reference' to another context manager in the header. The reference specifies both the information necessary to contact the responder and the credentials of the initiator. The former consists of the transport protocol, cryptographic plug-in, and the address to be used to contact the responder. The latter is a username and password, given the authentication mechanism chosen (See above, section 5.3.2.1). A single string is used to represent this information; the exact format is described later (See below, section 5.4.1.1).

The request can also specify three further optional parameters that control the synchronisation. Firstly, the synchronisation 'mode' can be specified. This specifies whether information should be updated at the initiator – 'updateLocal', at the requestor – 'updateRemote', or on both sides – 'sync'. Hence, the implementation allows the flow of information during the synchronisation to be controlled. The default is 'sync'. Secondly, the 'type' of data to be synchronised can be specified. The options are 'context', 'data', and 'all'. These correspond to synchronising contextual information, configurations, or both types respectively. This enables the context manager to retain a synchronised context database but different configuration or vice versa. The default is to synchronise 'all' types of information. Finally, it is possible to specify how any conflicts that arise during the synchronisation should be resolved. A conflict occur when concurrent occurrence with dissimilar values exist for a data item on the initiator and the responder. Such conflicts can be resolved by either using the initiators value – 'useLocal', the responder's value – 'useRemote', or by

retaining the existing value on the respective sides – ‘noChange’. The default is to retain the existing values.

Protocol

To perform the synchronisation a context manager follows a certain protocol.

To begin with the initiating context manager establishes what information is available. This information is represented using a synchronisation profile. The synchronisation profile contains the occurrences of all relevant data items, where the relevancy depends upon the type of data being synchronised. To determine what information needs to be exchanged, a synchronisation profile is required for both the initiator and the responder. Thus, the initiator requests a synchronisation profile from the responder and then builds a local profile themselves.

When both the remote and the local synchronisation profile are available to the initiation, the profiles are compared and a list is compiled with the data item occurrences that are missing locally, i.e. on the initiator. Then, provided the local database should be updated, the missing occurrences are requested from the responder and subsequently added to the local database.

Once the local database has been updated, the initiator then compares the remote synchronisation profiles again. This time, though, it is to establish which occurrences are missing on the responder. Assuming that the remote database should be updated, the initiator then sends the missing information.

Completing this protocol successfully will result in the two context managers being synchronised according to the specified parameters.

5.3.3.3. Extensions

To enable the context manager to include advanced and custom functionality the implementation supports the use of extensions. The extensions are executable units of code that are run inside the context manager. Despite being internal to the context manager information access from extensions is governed by the same privacy protection mechanism as external agents in the infrastructure. This ensures a subject’s privacy preferences are enforced.

With extensions it is possible to provide a wide variety of functionality. Later in this section two uses will be examined; integration with other infrastructures and data processing.

Operating modes

The context manager features two operating modes for extensions. First of all extensions can be run as background processes. In this mode the extension is invoked during the initialisation process of the context manager and is then run in a separate thread until either the extension or the context manager terminates. Running an extension as a background process can be useful if it is to perform actions at frequent intervals. Extensions can also be run on a per-request basis. In this mode an extension is only run for the period required to fulfil the processing of the incoming request. Extensions operating in this mode can be configured to be invoked on all requests or alternatively on certain specified requests only. It is also possible to configure whether the extension is to be invoked before or after the requests' actions have been performed. Furthermore, an extension can be set to be triggered on certain actions only, i.e. read/write/history. Hence, processing can be limited to relevant situations, and thus minimised. Finally, it should be noted that these two modes are not mutually exclusive. An extension can operate in both modes simultaneously.

Resource Extension Plug-ins

To ensure consistency extensions are implemented as plug-ins. These are referred to as resource plug-ins.

The resource plug-ins, just as others, must follow a set naming scheme. The scheme divides the class name into a plug-in name and a suffix. As previously, the plug-in name can consist of any combination of lower case letters. The suffix on the other hand is specific to the resource plug-ins and must be 'REP', in capital letters. Furthermore, the plug-ins must be located in the package 'net.osbakk.pi.cm.plugins'.

The plug-ins must also implement the 'REPInterface', outlined in Figure 53. The interface requires three methods to be implemented by the plug-ins. These are a method to initialise the plug-in, another to start the plug-in as a background service, and a method to invoke the plug-in on a per request basis.

To enable the resource plug-ins to communicate efficiently with the context manager, the interface contains a direct link to the context manager. The context manager link, 'CMLink', is provided as an initialisation argument. Full details of the 'REPInterface' and the 'CMLink' class can be found in Appendix C.8 and C.9 respectively.

Name: <plug-in name>REP†
Package: net.osbakk.pi.cm.plugins
Impl. interfaces: REPInterface
Methods:

init	start	perform

Notes: N/A
†<plug-in name> follows regular expression [a-z]+

Figure 53. Resource extension plug-in specification.

MobiComp plug-in

A plug-in has been developed that enable integration with the MobiComp infrastructure [Ryan 2005]. The mobicomp plug-in provides two way synchronisation between the two infrastructures. To achieve this, the extension utilises both operating modes available, retrieving information at periodic intervals and updating information on a per request basis. It is important to emphasis that both of these operations are performed without modifying the MobiComp infrastructure. The plug-in simply interacts with an already available webservice.

The MobiComp plug-in works as follows and its lifecycle is illustrated in Figure 54. When the context managers starts-up it initialises the MobiComp plug-in. During the initialisation the plug-in parses the provided arguments and properties extracting the address of the remote services and the credentials to use locally. The plug-in will also build the dictionary that will be used to translate messages between the two infrastructures. Once initialised the context manager starts the plug-in as a background service, assuming it has been properly configured. This thread will act as a timer, prompting the plug-in at a configurable interval to contact the MobiComp infrastructure to retrieve information. When the background services has been started the plug-in sits in a

waiting state until it is awakened by the timer, it receives a call from the context manager on the perform method, or the context manager terminates. A timer event will cause the plug-in to contact the webservice in the MobiComp infrastructure with a request for relevant information. If information is received, it is then processed and posted to the context manager using the CMLink and the credentials extracted during the initialisations. A call to the perform method prompts the plug-in to send information to the MobiComp infrastructure instead. It will first post a request to the context manager for the piece of context giving rise to the call. This is once again achieved using the CMLink and the extracted credentials. The information received from the context manager is then processed and sent to the webservice in the MobiComp infrastructure. Finally, terminating the context manager will also terminate the plug-in, stopping the background service.

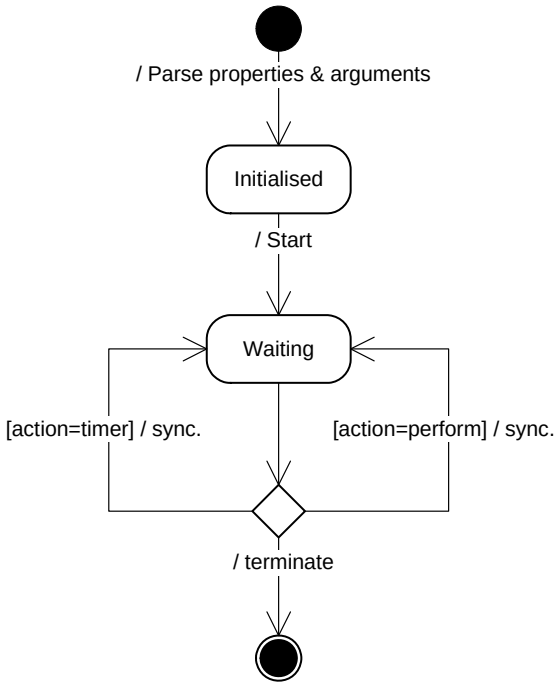


Figure 54. MobiComp plug-in lifecycle.

Data processing plug-ins

The extension mechanism provides the necessary framework for context managers to support data processing. Whilst implementation details are extension specific, the basic design of all data processing plug-ins is similar.

Firstly, as a result of the architecture, data processing can be performed at three different times. By configuring the plug-in to be invoked on write requests,

processing can occur when context information is stored. This is useful, for example, to validate incoming data. Similarly, if configuring the plug-in to be invoked on read requests, processing can occur when information is retrieved. This allows a just in time approach to data processing, for example transforming information before release. The last alternative is to configure the plug-in to run as a background process. This enables the plug-in to determine when processing shall occur, for example scheduling it at predefined intervals or during periods of low activity.

Secondly, context information is accessed by issuing messages containing read, write, and history requests. Thus, data processing plug-ins use the same mechanism to access information as external agents. Consequently, requests are subject to access control. However, for local information the plug-ins benefit from being able to use the context manager link to communicate. This removes the need for network connectivity improving availability of data and performance.

Finally, to perform the actual data processing the plug-ins can make full use of the features available in the platform, i.e. J2ME CDC/PP. In addition to the standard configuration and profile, plug-ins can also use any optional APIs and third-party libraries that are available to the runtime environment. Hence, plug-ins can take advantage of device specific resources to optimise processing. Furthermore, using network connectivity it is possible for plug-ins to distribute the workload to external resources.

5.4. Catalogue Service

The Context Manager Catalogue Service (CMCS) is an infrastructure component that simplifies the addressing in the infrastructure. There can be any number of CMCS in the infrastructure. Each CMCS maps Context Manager URIs to traditional URLs and operate independently.

5.4.1.1. Context Manager URIs

The primary address scheme implemented in the infrastructure is URI-based [rfc2396][rfc2718]. The basic structure of a URI consists of two parts: a scheme and a scheme specific part [rfc2396], as illustrated in Figure 55.

<scheme>:<scheme-specific-part>

Figure 55. Basic URI syntax

The name of the scheme to be used in the infrastructure is ‘cm’. This emphasises that it is a context manager that is being referenced. The scheme specific part is then used to encode the address information required. This includes both mandatory and optional parameters, some of which are dependant on each other.

The scheme specific part of the context manager URI is itself URI-based. The scheme specific part consists of two mandatory elements, one that specify the protocol and another that specify the address. It may also optionally contain user credentials, a reference to a context item, and a pointer to a particular occurrence of a specified context item. Note the use of the slashes that signify a hierarchal scheme. Figure 56 summarise the syntax of a CM URI.

cm:<protocol>://[<credentials>]<address>[<context-item>][<occurrence>]

Figure 56. CM URI syntax

The protocol element specifies the transport plug-in and the encryptor decryptor plug-in. Together these therefore indicate what is required to communicate with the context manager on the address in question. In the protocol element the names of the two plug-ins are separated using a dash. For example, if the socket transport plug-in and the RSAAES encryptor decryptor plug-in are to be used the protocol element would be ‘socket-rsaaes’.

The details of the address element are dependant on and specified by the transport plug-in being used. It can consist of any combination of alphanumeric, unreserved, and reserved characters [rfc2396]. If the address contains reserved characters it must be enclosed within square brackets. For example, the socket transport protocol takes an IP-address or a domain name as an address.

The optional credentials in the current implementations specify the username to use when accessing the context manager. It can also specify the password, though this is not recommended. If provided, the password will be removed from the URI and sent within the message header when communicating with the

context manager. If both username and password is specified then they are separated using a colon. The credentials must be followed with an at-sign.

The optional context-item element points to a particular context item at the context manager being referenced. The hierarchical path to the context item is encoded as a string that is delimited by periods, i.e. the same technique used when referencing items within a context profiles. A context-item element must be preceded with a hash. For example, to reference a user's location the pointer '#location.coordinates' could be used.

The optional occurrence element can be used to reference a particular instance of a context item. Hence, the occurrence element can only be used in conjunction with the context-item element. An occurrence is specified as the difference in milliseconds from midnight the 1st of January 1970 coordinated universal time. (UTC). The element is specified using digits only and must be preceded by a colon.

Figure 57 shows an example of an address with all the optional elements included. The CM URI specifies that the socket transport protocol and the rsaaes encryptor decryptor plug-ins should be used. It also states that the context manager is located on the localhost. Furthermore, it specifies that the username and password that should be used are 'user' and 'pass', respectively. Finally, the context item of interest is 'location.coordinates' occurring 12th of September 2005 at 03:35:11.

cm:socket-rsaaes://user:pass@127.0.0.1#location.coordinates:1126488911740

Figure 57. Example of a CM URI

5.4.1.2. The service

In the infrastructure the CMCS is implemented as a webservice using Java Servlet technology [Java Community Process 2001]. The CMCS's creates a mapping between one or more CM URIs and a static well known URL. This serves two important functions. First of all, it provides the ability to map dynamic addresses onto static and well known addresses. Secondly, it makes it possible to reference entities without specifying the underlying transport plug-in(s) and the encryptor decryptor plug-in(s). Hence, the CMCS plays an

important role in making the use of unreliable communication feasible over an arbitrary number of protocols and mediums.

At the heart of the CMCS is a directory services, allowing an agent to lookup the context managers associated with a listed entity and their respective CM URIs. To aid the agent in determining which context manager to use when alternatives exist, each context manager is indexed with an integer. The recommendation is for an agent to contact the context manager with the lowest index possible. Each context manager listed may contain several CM URIs with which they can be contacted. To minimise redundancy, the CMCS list the URIs and ciphers separately for each context manager. If no context manager is associated with an entity or if all services associated are offline, the service will assume the entity is offline.

The information listed by the CMCS is automatically updated by the appropriately configured context managers. During the start-up the context manager will query each of the available transport server plug-ins about their address. The context manager will also compile a list of the available encryptor decryptor plug-ins. This information is then communicated to the CMCS that then can update the listing. The context manager will also update the CMCS when it is being shutdown to notify the services it will go offline and be unavailable.

Finally, the CMCS also provides a simple web interface for user management. The interface allows the context managers using the services to be listed along with their entries. It also allows a user to update their password, i.e. the password used by their context manager to authenticate themselves to the services. Furthermore, the CMCS's administrator has the ability to use the interface to add and delete users.

5.5. Proxies

A Proxy is a component type not previously described. In the infrastructure it is any component through which information flows but that is neither the origin nor destination of the flow. From the point of view of the information source the proxy will appear as the destination and from the destination the proxy will

appear to be the source. Proxies are used to process data in transit or to relay messages. This enables otherwise incompatible services or unreachable destination to be integrated with the infrastructure.

5.5.1. Proxy types

The first and simplest type of proxy emulates a destination with which there is a one-to-one relationship. This enables the proxy to be used without requiring any special support by the infrastructure. The type of proxy must only adhere to the interfaces a context manager and agent use. In this form the proxy will be personal, i.e. serve only one entity. Whilst this implies that the usage is restricted, there are situations where a personal proxy is preferable. For example, this is the case when sensitive information is to be processed. A personal proxy, over which the subject has control, provides a trustworthy alternative for the processing.

The second type of proxy supported by the infrastructure uses explicit routing instructions. This allows it to support multiple destinations. Hence, it does not require any special relationship to exist between the proxy and the destination, i.e. the proxy can be shared. To facilitate this second type of proxy in the infrastructure two additional header attributes are used. First of all, the source must specify the next destination after the proxy using the ‘relayTo’ header attribute. The destination could be the final destination or the address of another proxy. Secondly, the source has the option of adding the message to be relayed in a ‘relayMessage’ header attribute. This allows the source to fully specify the message being relayed including how it is encrypted. If the ‘relayMessage’ attribute is not present then a proxy will process and relay the incoming message. In addition to these two attributes each proxy can support, and even require, additional custom header attributes to be used. Furthermore, it should be noted that even though no special relationship is required between the proxy and the destination an implementation can limit the destinations allowed.

5.5.2. Examples of usage

From the point of view of the infrastructure the proxy component type is a tool with which external functionalities can be incorporated into the infrastructures. For what purpose this tool is used depends on service providers and users. The

possibilities are there to develop a wide range of different proxies. Three examples of how a proxy component can assist the infrastructure are translation between protocols, the anonymisation of communication, and data transformation.

5.5.2.1. Protocol translation

Under some circumstances it may be beneficial, or necessary, for different transport protocols to be used by an agent and a context manager wishing to communicate. For example, assume a user wants to minimise the number of protocols the context manager running on their mobile devices support without reducing the compatibility with agents. A protocol translation proxy can then be used in the infrastructure to assist the mobile context manager.

For the infrastructure a protocol translation proxy has been implemented that forms a bridge between two transport protocols, HTTP and Socket. The proxy has been implemented as a Java Servlet [Java Community Process 2001] that emulates a context manager. Hence, one-to-one relationships exist between the proxy and an entity. It accepts requests from agents over the HTTP transport protocol. These requests are then forwarded to their destination over the socket transport protocol, e.g. to a user's mobile context manager. Thus the context manager only needs to accept communication over one of the two transport protocols, namely the socket. To include the HTTP transport protocol in the updates sent to the CMCS a dummy HTTP transport server plug-in is used. The dummy plug-in makes the context manager aware that it can be contacted using HTTP and allows the address of the protocol translation proxy to be set. Note that the dummy plug-in does not set up any HTTP server, nor does it contain any such implementations. It simply provides the necessary information needed when updating the CMCS. Figure 58 illustrates the use of the protocol translation proxy.

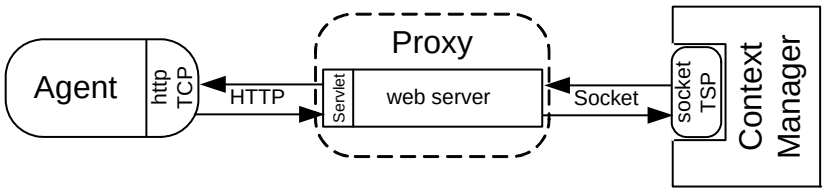


Figure 58. Using a protocol translation proxy

5.5.2.2. Anonymising communication

The aspect of anonymous communication has previously been described (See above, section 4.5.3). The idea is to break the path between the communicating parties. Proxy components enable this to be achieved in the infrastructure. Hence, using proxy components anonymous communication solutions can be developed. An architecture has been designed that shows how such a solution can look.

In the architecture proxy components are used together with an anonymity network. The proxy components provide the entry and exit points to the anonymity network. This separates the implementation of the anonymity network from the infrastructure, but still allows the network to be used as an external component. The anonymity network's task is to obfuscate the message path. For the purpose of this thesis it is assumed the anonymity network performs it tasks efficiently and securely. Hence, after a message has reached the entry point it is the responsibility of the anonymity network to route the message to its final destination.

5.5.2.3. Data transformation

Proxies provide a compliment to the support for data processing available with the extension mechanism in the context manager (See above, section 5.3.3.3). The advantage of using proxies is that they enable data to be processed in transit, rather than at the destination. This makes them particularly useful for transforming data.

For example, take a situation in which basic positioning information is required in the form of longitude-latitude coordinates. Then assume that a context producer supplying NMEA data from an associated GPS device is available. From the NMEA data the current coordinates can be extracted. By using a proxy for the processing and placing it between the context producer and the context manager, a seamless transformation from NMEA data to longitude-latitude coordinates can be achieved.

Whilst it would be possible to obtain the same end result by using a data processing extension, it is not as efficient as using a proxy since it requires the NMEA data to be stored in the context manager for processing. It produces

redundant information as well as introducing an additional write and read cycle. Hence, for the given situation the use of a proxy is more appropriate.

The use of proxies for data transformation, and processing in general, also benefits from the fact that proxies can be hosted anywhere in the network. Hence, operations that require special resources or are demanding can be offloaded. In the example above, where NMEA data is to be transformed into coordinates, this would not normally be the case. However, there are certainly situations where resource constrained devices are unsuitable or cannot perform desired transformations. Generally, the processing of images, video, and sound would be examples of such cases.

5.6. Agents

A number of agents have been developed to use the infrastructure, including an iButton Context Capture application for gathering information, a Context-Aware Desk Display that shows information about the occupier, and a Web Presence application for publishing context information.

5.6.1. Administration console

To interact with the context manager an administrative console has been implemented. The console takes the form of a standalone application that communicates with the context managers, as illustrated in Figure 59. It can be used to interact with both local and remote context managers.

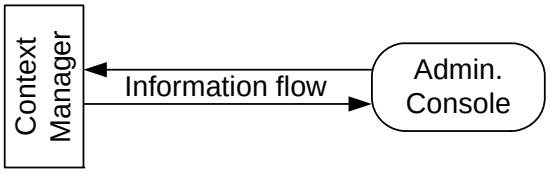


Figure 59. Administration console component.

The console features a basic graphical user interface that allows various aspects of the context manager to be configured, see Figure 60. First of all the application provides the means for users to set up their privacy preferences using the role based access control mechanism. Using the console, lists of access controls and roles can be defined, users created, and ruleset loaded. See Appendix D for an example. Secondly, the administration console enables resources to be defined, configured, and bound to context actions. Thirdly, the

application provides the ability to perform system commands. Commands are available to enable the user to synchronise two context managers and to shutdown the connected context manager. Finally, the console also provides limited means for users to manually view and change contextual information. A useful feature, available to the subject when viewing their own context, is the ability to utilise the application as a form of vampire mirror [Butz, Beschers et al. 1998] displaying the information disclosed with respect to registered users.

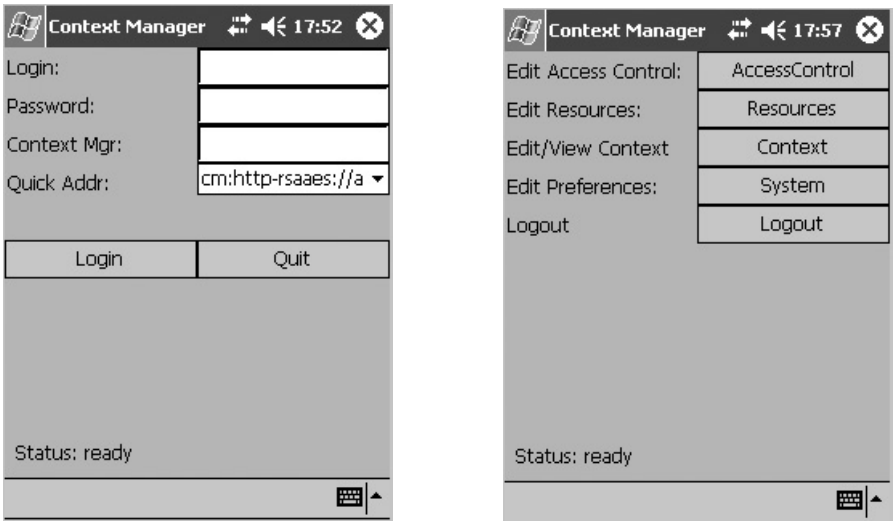


Figure 60. Screenshots from the administration console.

5.6.2. Context-aware desk display

The Context-Aware Desk Display consists of an enclosed TINI (Board Model 390) microcontroller [Loomis 2001] with an attached LCD and keypad, see Figure 61. The display has been designed with the ordinary name tag, sometimes found on desks, in mind. But instead of simply displaying the name of the person occupying the desk, the display also presents additional context information like the person's email address, if they are in today or when they are expected to be in next. The information presented by the context-aware desk display originates from the person's context manager and the application can easily be extended to display more information.

Figure 61. Context-aware desk display.

The context-aware desk display has been designed to utilise the extensions-architecture of the context manager. It uses a resource plug-in to detect relevant context changes and send updates to the display device. On the display device, i.e. the TINI microcontroller, a webserver with a servlet container is run. Deployed on that webserver is a desk display servlet that receives context updates and drives the LCD. Figure 62 illustrates the use of the context-aware desk display component.

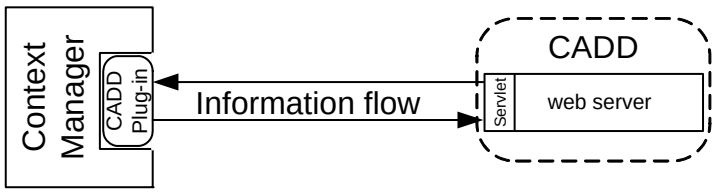


Figure 62. Context-aware desk display component.

Due to the privacy mechanism in the context manager the resource plug-in cannot retrieve any information by default. Consequently without further action no information will flow to the display. Therefore, to activate the context-aware desk display the user must grant the plug-in read access to location.place, user.business-info.next-in, and user.home-info.contact.online.email. This is achieved by using the administration console to create a new user account for the plug-in and defining a context-aware desk display role with associated access control list(s) containing the appropriate permissions. Should the user later decide to revoke access, e.g. by removing the account, the information flow to the display is stopped.

5.6.3. Web presence application

It has long been the vision of projects such as Cooltown [Kindberg, Barton et al. 2002] that every entity should have a web presence. The web presence application provides this for the entities using the infrastructure. Although the context manager itself can be seen to provide a form of web presence, it is not directly accessible by standard web browsers. The web presence application provides the link between a context manager and a web browser, as illustrated in Figure 63.

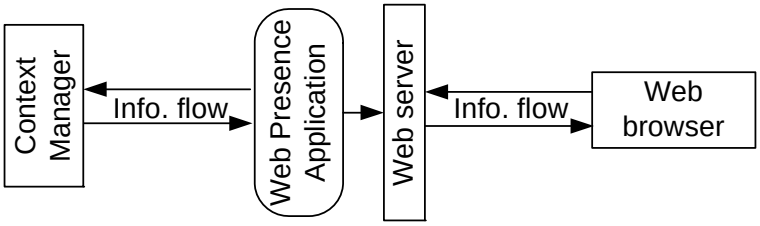


Figure 63. Web presence application component.

The application has been designed to handle a high work load without affecting the performance of the context manager. It has therefore been implemented to provide a snapshot view of an entity's context using existing web technologies. The information that makes up the snapshot is gathered by the application at regular intervals by requesting all the context information available to it. This information is then translated into html and/or javascript, formats compatible with web browsers. Once translated, the publication of the context information is then handled by a standard web server.

By default the web presence application is not allowed to retrieve any information from the context manager. Hence, the application must be granted read access to the elements that should be published. For example, if the entity's activity is to be published the application needs access to activity.type and perhaps activity.note. In general, a separate user account for the web presence application therefore needs to be created using the administration console. The account also needs to be assigned any roles necessary to provide the application with access to the desired information. If at a later stage access is changed or revoked, this affects what information is published at the next update interval.

5.6.4. iButton context capture application

The iButton context capture application uses iButtons [Dallas Semiconductor, Maxim 2005B] to gather context information. iButtons are small, uniquely identifiable, devices. They may contain memory or have abilities like temperature sensing. iButtons may be identified and their memory content or sensor values read by a PDA with an adapter [Osbakk 2006] as well as more novel mechanisms [Laerhoven, Schmidt et al. 2002]. Figure 64 shows an iPAQ equipped with an adapter for reading iButtons.

Figure 64. iPAQ equipped with an iButton reader.

The unique identities of iButtons allow context information to be logically linked with physical tags. This is achieved by keeping a database of known tags, together with their associated meaning. In this way iButtons can be used to tag context information including locations, objects, and even activities. Thus by reading iButtons context information can be captured.

The iButton context capture application makes use of the possibility iButtons provide for the capture of context information. The application has been implemented as a standalone program to be run on PDAs. It uses a local database to hold tag information. The tag information is user-specific but a generic set may be distributed with the application. Every time a tag is read the database is consulted and the appropriate context is deduced. The application then proceeds to update the information held by the entity's context manager(s). To read tags this application has been implemented to use an adapter [Osbakk

2006] as well as two manual fall back modes. When the adapter is used the application is first placed in a reading mode. After this the adapter's connector is simply touched against the iButton tags to be read. The two manual modes, on the other hand, require the user to first bring up a list with either tag names or tag identifiers. Then it relies on them correctly selecting the tag to be 'read' from the presented list. Whilst the manual modes require more work to be done by users, they do offer alternatives to those without access to adapters. Figure 65 shows two screenshots from the application.

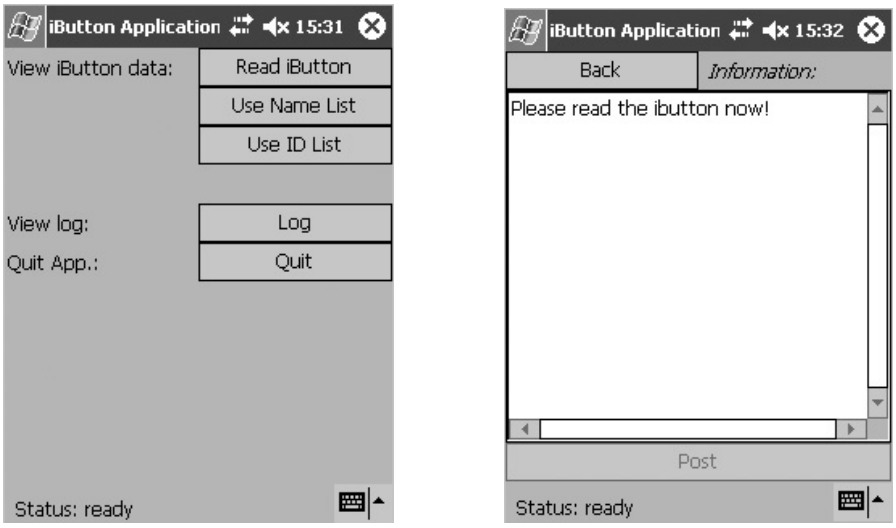


Figure 65. Screenshots from the iButton context capture application.

As a context producer, the iButton context capture application must be able to store information in the entity's context manager to function. By default the privacy mechanism prohibits access. Thus, the application must explicitly be granted write access to context elements linked to the iButtons tags. For example, if tags are used to denote room locations the application needs write access to location.place. Hence, a user account must be set up as well as an appropriate role and its associated access control list(s). This is achieved using the administration console. The granted access can be modified or revoked at any time, with changes taking effect immediately.

5.7. Summary

In this chapter the implementation of the developed privacy enhancing infrastructure has been presented.

As stipulated by the design in Chapter 4 the implementation consists of several components including a context manager, a catalogue service, data storage, and agents. The targeted platform for the implementation of these components has been the Micro Edition of the Java 2 platform, with the Connected Device Configuration, the Foundation Profile, and the Personal Profile.

The main component in the infrastructure is the context manager. Much of the functionality has therefore been developed in conjunction with it including communication, privacy protection, and request fulfilment. To attain the desired modular design plug-ins are used to both enable and secure communication. The implementation includes plug-ins for socket-based communication and RSA-AES cryptography. Plug-ins is also used to extend the functionality of the context manager, for example a plug-in has been developed that integrates this infrastructure with another. Furthermore, the plug-in architecture provides the necessary framework for supporting data processing.

The communication with the context manager is request-based, where interaction dialogues are split into sequences of messages. Each message is individually processed. This includes authenticating the user using a username password, determining access using the implemented role based access control mechanism and P3P, and fulfilling valid requests.

A number of agents has been presented that interact with the context manager including an administration console that allow a user to interact with their context managers, a context-aware desk display that provides information about a user's presence, a web presence application that publish contextual information to the web, and an iButton context capture application that allows information to be gathered. Together these applications aim to demonstrate that the infrastructure works and is feasible to use.

How the context manager catalogue service simplifies the addressing in the infrastructure has also been described. As presented the infrastructure

implements a URI based addressing scheme. The scheme enables the address of a context manager be specified along with the transport plug-in and cryptographic plug-in to be used with a single Context Manager URI. As an entity can have multiple associated context managers, each with several addresses, this results in each entity having many CM-URIs. The implementation of the context manager catalogue service makes this transparent by mapping an entity's CM-URIs to standard a URL using Java Servlet technology.

Introduced with the implementation is also another type of component, namely proxies. Proxies sit between the source and destination in a message path and enable in-transit processing. In the infrastructure a protocol translation proxy has been implemented. It has also been described how proxies can be used to anonymise communication and transform data.

CHAPTER 6

EVALUATION

The aim of this work has been to improve the overall level of privacy enjoyed by users in context-aware environments and to make the development of applications quicker and easier. The hypothesis made was that this could be achieved through the use of a dedicated privacy-enhancing infrastructure for context-awareness. The work has gone on to develop and implement such an infrastructure as described in the two previous chapters.

This chapter evaluates the implemented infrastructure to determine the correctness of the hypothesis. The evaluation first of all establishes to what extent the privacy protection meets the captured requirements and what the resulting level of control is. It then looks at the development support provided to establish if the infrastructure makes the development of applications easier and more rapid. Finally, the evaluation compares the approach taken in this work with related work.

6.1. Privacy protection

Evaluating privacy protection is not an easy task. It is complicated both because the perception of what constitutes a privacy violation is subjective and because privacy is an abstract concept that is difficult to measure.

An action that one person finds intrusive can be acceptable to others. Hence to achieve an accurate understanding of how well a subject's privacy is protected, each case must be individually analysed and compared against that subject's preferences. Performing this type of individual analysis is not only time consuming, but also brings issues of its own. For example, given that each case comes with an individual frame of reference, i.e. the subject's preferences, how then can cases be compared and data be aggregated?

This brings us to the second issue, that of measuring privacy. Privacy, unlike performance for example, does not feature a common scale by which it can be accurately measured. Hence privacy has no associated units in which it can be expressed. Furthermore, similar intrusions cannot be directly compared between subjects. The 'cost' of each privacy violation, or the 'value' of its corresponding protection, is individual and dependant on many factors.

Given these issues, the approach taken in this work has therefore been to assess the degree to which the privacy requirements are met and the abilities the employed privacy protection mechanism provides to describe privacy preferences.

6.1.1. Requirement fulfilment analysis

There were five privacy requirements captured (See above, section 4.1.2), each covering a different aspect related to privacy.

- Decentralised structure
- Retain offline level of privacy
- Customisable effective balance of privacy
- Handle known and unknown recipients
- Security

Each of these requirements will now be analysed.

6.1.1.1. Decentralised structure

The chosen design of the infrastructure upholds this requirement (See above, section 4.4). There are no central authorities in the infrastructure, each context manager operates independently. The scope of the infrastructure is customisable. A user may choose to only allow interaction with their own subset of components or they may choose to cooperate with others. Either way the infrastructure will work the same. There are however components that can be argued to be centralised like the Context Manager Catalogue Service and Proxies.

The CMCS provides the translation between virtual, but static, addresses and the actual addresses at which context managers can be reached (See above,

section 5.4). To do this it is necessary for the CMCS to be located at a fixed, known, address and be continuously operating. As such it is expected that the number of catalogue services will be limited, and then arguably be centralised. However, the use of the CMCS is not compulsory. Furthermore, a user may choose to operate their own catalogue service without incurring any penalties. Thus whilst a centralised CMCS can be used, it is at the discretion of the user.

A similar reasoning can be used for proxies. Proxies are placed in the message communication path. Their job can be to provide a bridge between different protocols or to transform the message content (See above, section 5.5). Whilst there are benefits from using centralised proxies, it is not necessary that they are operated in this fashion. Furthermore, the use of proxies is not compulsory. Thus centralised proxies may be used but it is once again at the discretion of the user.

6.1.1.2. Customisable effective balance of privacy

It is required that the effective balance of privacy is customisable. This implies that the infrastructure must provide the subject with the means to influence their privacy, within the range specified by society. To reflect the scope of control in the privacy model, the requirement was broken down into three parts.

First of all, an access control mechanism was found to be required with which a subject is able to control their own disclosures and receptiveness to information. In the infrastructure the context manager is responsible for providing this mechanism (See above, section 4.4). To fulfil the requirement the context manager implements a role-based access control mechanism (See above, sections 4.5.1 & 5.3.2.2). The extent to which the implemented mechanism is customisable to the user's preferences is however limited. Since the focus has been on static privacy preferences only, there is currently no support for context sensitive access controls. Furthermore, since each role, when resolved, consists of a set of individual access controls it is not trivial to scale up or down the overall access granted (See above, section 4.5.2.2).

Secondly, a mechanism for achieving anonymity was required to allow a subject to hide their identity. Even though the context managers and the agents in the

infrastructure have been designed to not be directly identifiable, certain 'features' can be used to identify them. For example, context managers can be identified by the context information they contain, the address information used to contact them, and if cryptographic plug-ins are used from their public keys*. Similarly agents can be identified from the credentials they use, by the address from which requests originate, and from their public keys. To tackle these issues a number of counter measures can be used:

1. The context managers can use the implemented access control mechanism, assuming it is correctly set up, to hinder identifying information from being released.
2. Agents can choose to use the username 'anonymous' to post requests without revealing their identity.
3. The context managers and agents can regenerate their public keys at frequent intervals and only use dynamically changing addresses to avoid being identified by them.

These counter measures provide a basic level of protection against identification. However, they are neither optimal nor sufficient. To, for example, rely on the assignment of varying dynamic addresses to attain anonymity is not recommended. Furthermore, the regeneration of public keys is time consuming and still allows the subject to be tracked within the key cycles.

To provide anonymity, the infrastructure has been designed to use a proxy. By tunnelling incoming and outgoing communication through a trusted anonymising proxy the identity of the subject is protected. The drawbacks of this approach are that the proxy must be trustworthy and that it must be centralised to some degree. Trustworthiness is important because the proxy will be aware of the true identity of the subject. It may also see the contents of requests if they are to be transformed from being encrypted with one key to another. The requirement of being central is necessary for the proxy to be able to provide a good level of protection. A proxy that is only used by a single

* Note even if the keys do not contain any identifying information or are guaranteed to be unique, they are considered to provide a reasonable means of tracking a subject over time.

subject is of little use. With these mechanisms the infrastructure design meets the requirement of anonymity. It should be noted that only the basic-level of protection has been implemented, as of yet.

Finally, a mechanism that allows subjects to use pseudonyms was required. This mechanism should enable a subject to be identified with a manufactured identity, with which they cannot be linked. It was also required that the manufactured identity, i.e. the pseudonym, could be retained over time. This requirement is similar to that of providing anonymity. However, the latter requirement makes it impossible to simply rely on the identifiable information to be untraceable. Hence the basic protection mechanism that renews identifying information such as public keys and addresses is not enough.

To fulfil the requirement the infrastructure has therefore been designed to use proxies. As previously described proxies can be used to anonymise requests. A similar procedure is used to provide a subject with a pseudonym with the difference that the projected identity will remain fixed over time. Hence the recipients of the request will perceive them coming from the same identity, but do not know their true origin. When desired the subject could of course change the used pseudonym, either by changing the settings of the proxy or swapping proxy altogether.

The drawbacks of the pseudonymity-proxy are the same as those for the anonymity-proxy; it needs to be trustworthy and centralised. With the pseudonymity-proxy the infrastructure design is able to also meet the requirement of pseudonymity. The implementation does however not feature pseudonymity yet.

6.1.1.3. Retain offline level of privacy

To establish if the requirement to retain the offline level of privacy is fulfilled it is necessary to return to the conceptual model of privacy (See above, section 3.2). The model presents four ways in which a user can control their privacy offline (See above, section 3.2.3). How well the infrastructure handles each of these will now be established.

First of all, the model states that subjects are able to control their own disclosures. In the infrastructure this corresponds to controlling the access to information held in their context manager. The context manager does provide a mechanism with which access can be controlled (See above, sections 4.5.1 & 5.3.2.2). There are, however, limitations to the extent this mechanism is customisable (See above, section 4.5.2.2).

Secondly, the model states that subjects can control their presence or recognition in public spaces. In the infrastructure this corresponds to controlling the presence of their context managers and agents in the infrastructure and whether or not they are possible to identify. The former is possible in the infrastructure since the subject is able to control whether they run any components or not. The latter concerns the possibility for the subject's components to remain anonymous. This feature is also supported by the infrastructure design and has been evaluated in Customisable effective balance of privacy (See above, section 6.1.1.2).

Thirdly, the model states that a user can control their actions and links therewith in public spaces. In the infrastructure this corresponds to the ability to control their context managers' and agents' actions and their association with these components. The former is possible in the infrastructure as all actions performed by the subject's context managers will be following either their privacy preferences or the resource configuration they have set up. Hence the context manager will not act autonomously in its default configuration. And even though it is possible for a subject to add autonomous behaviour to the context manager using resource plug-ins, this is then considered to be a conscious choice made by the subject. Similarly, the subject is able to control the agents they deploy in the infrastructure and their configuration. The control of the latter, their association with deployed components, is also supported by the infrastructure design through the use of pseudonyms. This feature has also been evaluated in Customisable effective balance of privacy (See above, section 6.1.1.2).

Finally, the model states that subjects are able to control their receptiveness to information. In the infrastructure this corresponds to controlling the ability to

store information in their context manager. This is achieved using the same mechanism that controls disclosures.

6.1.1.4. Handle known and unknown recipients

It is a requirement that known and unknown recipients should be handled alike. The requirement thus states that it must be possible to express privacy preferences, form agreements, and disclose information independent of whether the identity of the recipient is known or not.

These requirements are fulfilled by the infrastructure through the use of the Platform for Privacy Preferences (P3P) (See above, sections 4.5.2.3 & 5.3.2.2). The mechanism implemented in the infrastructure exploits P3P policies and rulesets to form the agreement between the subject and the requester without knowing their identity. The rulesets are defined by the users and capture their privacy preferences with respect to the usage of information. The P3P policies are defined by the requester and present their intended use of the information, the contract on which information may be released. By matching rulesets and policies the appropriate information can be disclosed.

Whilst the use of P3P fulfils the stated requirements, it has its limitations. As previously discussed there is no technical mechanism available with which an agreement can be enforced or P3P policies verified (See above, section 4.5.2.3). The lack of control over the use of disclosed information is however not limited to the P3P mechanism. Any mechanism that releases information in a understandable form is open for abuse. This is why the privacy model emphasises trustworthiness in the process of disclosure (See above, section 3.2.4). Another limiting factor is the lack of tools to support the creation of both rulesets and policies. There are 11 references listed at the standards page to compatible P3P editors, generators, and checkers [World Wide Web Consortium 2005B]. Out of these, five policy editors and one ruleset editor are available, see Figure 66. The majority of the available policy editors are web-based and do not support any extension to the vocabulary. Two are applications, of which one has been confirmed to work with the required extension. The ruleset editor is an application and has been confirmed to support the vocabulary extension. Thus the choice of what tool to use is limited.

	Type	Format	Extension
A) P3P Builder	Policy editor	Web-based	Not supported
B) P3P Edit	Policy editor	Web-based	Not supported
C) P3P Writer	Policy editor	Web-based	Not supported
D) P3P Policy Ed.	Policy editor	Application	Supported‡
E) P3P Editor	Policy editor	Application	Unknown†
F) JCR P3P …	Ruleset editor	Application	Supported‡

† Unconfirmed due to unavailability. ‡ Requires custom post-processing.
A) [P3P Builder 2005] B) [Code Infusion 2005]
C) [P3P Writer 2005] D) [IBM alphaWorks]
E) [Abrantix 2005] F) [JRC P3P Resource Centre 2005B]

Figure 66. Table with available P3P policy and ruleset editor tools.

Overall it can be concluded that although the P3P mechanism fulfils the captured requirement, the solution is not optimal. For the use of P3P to be practical the support for the generation of both policies and rulesets needs to be improved. Furthermore, a mechanism also needs to be developed that deters misuse.

6.1.1.5. Security

The infrastructure is required to be secure to attain an acceptable level of privacy. Without adequate security even the most elaborate privacy protection mechanism will fail. The requirement highlighted five aspects of security, of which anonymity has already been evaluated in the section Customisable effective balance of privacy (See above, section 6.1.1.2). The remaining aspects: authentication, confidentiality, integrity, and availability will now be discussed.

In the implementation authentication is performed using a username-password mechanism. This is the simplest of the candidate mechanisms (See above, section 4.5.1). As discussed earlier its susceptibility to replay attack has been deemed to be an acceptable weakness for the purposes of this work (See above, section 5.3.2.1). However, the difficulty concerning key management is a problem. Each entity to entity relationship requires a shared secret. To ensure security this secret must be difficult to guess, it should therefore not be reused by the entities in other relationships. As a result scalability becomes an issue.

To ensure the confidentiality of communication cryptographic plug-ins are used. With the infrastructure a RSAAES plug-in has been developed that uses a combination of an asymmetric and a symmetric cipher to secure communication (See above, section 5.3.1.3). Using this plug-in the security of the communication will be a function of the key and its size. In the infrastructure implementation pseudorandom keys are generated with a default key size of 1024-bit and 256-bit for the RSA and AES components respectively. For the purpose of this infrastructure this is deemed to be secure enough. However, whilst the use of the RSAAES plug-in is recommended, it is not mandatory. The confidentiality of communication can therefore vary with configurations. It is always possible, though, for each entity to specify their acceptable set of communication plug-ins, thus prohibiting unencrypted communication.

The integrity of communication is not explicitly covered by the infrastructure. No mechanism for checking the integrity of messages has been implemented. However, when the RSAAES plug-in is used, to ensure the confidentiality of communication, basic integrity is implicitly provided. For the purposes of this work it is deemed difficult enough to tamper with the encrypted data stream without raising suspicion of the decrypted message. To be able to guarantee the integrity of messages to a specified level, further work is required.

With respect to availability attacks the protection provided in the infrastructure is very limited. It focuses on minimising the workload of processing unauthorized requests. The filtering process, performed as a part of the access control (See above, section 5.3.2.2), only lets authorised actions through to request fulfilment. The implementation also delays replying to unauthorised requests on purpose to make brute force attacks less favourable. Whilst these actions can protect the availability of the infrastructure when subject to infrequent, unintentional, attacks it is not feasible to expect them to protect against persistent and calculated attacks.

6.1.2. User survey

From the start it was recognised that the candidate access control mechanisms had limitations (See above, section 4.5.2). To evaluate the actual performance of the two primary privacy protection mechanisms presented in this work (See above, sections 4.5.2.1 & 4.5.2.2) a user survey was undertaken.

The survey consisted of a series of questions and tasks, found in Appendix E.1. To encourage participation the survey was performed online and was anonymous. Hence, no identifying information about the subjects was recorded. It is possible though, to distinguish between individual surveys through the use of the completion time. To gather participants the survey was advertised in newsgroups[*] and an online discussion forum[**].

6.1.2.1. Completion process

First of all the subjects were asked to complete a few questions about their background. This included questions about their age, gender, computer literacy, and internet use. These are questions that can be found in other surveys as well [GVU's WWW Surveying Team 1998]. This survey then specifically asked the users about their privacy concerns both with respect to their real-life and their presence online.

Then to evaluate the access control models the subjects were presented with a scenario. The scenario asked them to envisage that their mobile phone was connected to a context-aware system. The subjects were told that the system would be able to collect and monitor their context with location and current activity given as examples. They were also told that the information collected would be used to provide enhanced services; examples given included call management, navigation to points of interest, and context sharing.

After this the subjects' privacy preferences were captured to establish a baseline against which the setup of the access control mechanism could be compared. This was achieved by asking the subjects to indicate who they felt should be

[*] ukc.misc, ukc.comp-postgrad, alt.privacy

[**] http://www.ntcompatible.com/thread27503-1.html

able to access information about their location, activity, and contact details respectively. The alternatives they were given included family member, friend, colleague, boss/supervisor/teacher, and anyone (public). Multiple alternatives could be selected.

Once the subjects' privacy baseline had been captured the subjects were asked to set up a simulation of the CCS and RBAC mechanisms to reflect their privacy preferences. The simulations worked the same as the real access control mechanisms but were limited to the pieces of context and recipients relevant to the scenario and did not provide any feedback. To aid them in performing the set task, the subjects were given some brief information about the respective access control mechanism as well as instructions on how to set them up. Once they had set up the mechanisms they were also asked to indicate how similar they found the mechanism to be to how they reason about privacy day to day, how accurately they felt they could represent their privacy preferences, and how easy they found it to express their preferences using the mechanisms.

Lastly it should be said that no feedback was given during the completion of the survey nor once it had been completed. Hence the subjects were not told how accurately they managed to set up either mechanism.

6.1.2.2. Findings

The survey received 31 responses, compiled in Appendix E.2. The responses came from subjects in all of the following age groups: ≤20, 21-40, 41-60, and ≥60 years. A majority though, fell within the bounds of the 21-40 group. The ratio between female/male responses were 9/22. Furthermore the sample showed a bias towards computer and internet literate people.

In this sample there were some interesting findings both with respect to the subjects' privacy concerns and their use of the access control mechanism [Osbakk, Ryan 2004B].

Privacy concerns

It has been assumed that the ideal level of privacy for ubiquitous computing systems was the same as for offline environments (See above, section 3.2.2).

The responses on the survey however, indicate that this may not always be correct.

The sample shows that 97% of the respondents are either very concerned or concerned about their privacy when online. The corresponding concern for real-life, i.e. offline, privacy in the sample is 71%. Hence, the sample shows a clear difference between privacy concerns online and offline. This finding is illustrated in Figure 67.

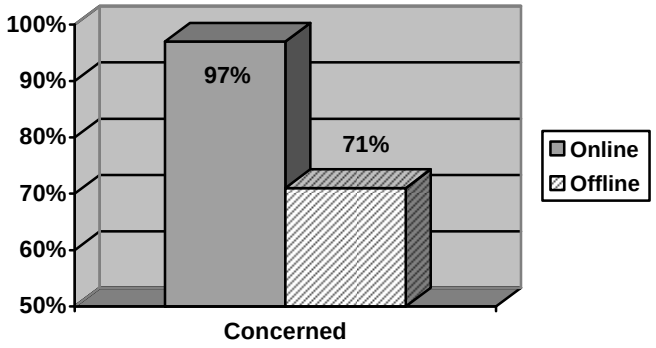


Figure 67. Proportions of subjects that reported privacy concerns.

The cause of this heightened concern is still unclear, though we have identified two hypotheses. It could perhaps be that people fear the unknowns that are encountered with new technology. Indeed, it is only within the last decade that Internet technology has become widely available to households [National Statistics 2005]. The greater concern could also be an indicator of a lack of trust in online technology, its providers, and the way personal information is handled. Given the surge of spam [Wakefield 2005] and other malicious online activities such as computer viruses [Ward 2005] and internet fraud [BBC News 2005], a lack of trust from those on the receiving end should not be surprising. Even those that are lucky not to be targeted are likely to be affected by the reports in the media.

Regardless of what has caused the subjects to show more concern for online than offline environments, this finding does not bode well. It suggests that the opportunity to form an early relationship of trust between the technology, its users, and providers has already passed. This can, potentially, negatively affect the acceptance and desire for ubiquitous computing by a wider audience. Hence, users are rightfully expected to inspect new ubiquitous systems harder than their

offline counterparts, if any. They may possibly also judge them harder than traditional systems, which is unfortunate. Overall, the finding suggests that the work of proving the trustworthiness of these new ubiquitous systems will at the very least be more difficult than anticipated.

Access control mechanisms

The responses to the survey also yielded some interesting findings concerning the tested access control mechanisms.

When comparing the setup of the Classification and Clearance Scheme (CCS) against the subjects' reported privacy preferences the average accuracy was found to be 79% in the sample. The respective average accuracy for the Role Based Access Control (RBAC) mechanism was found to be 87%. Hence the subjects in the sample were able to set up the RBAC mechanism more accurately than the CCS mechanism. This finding is illustrated in Figure 68.

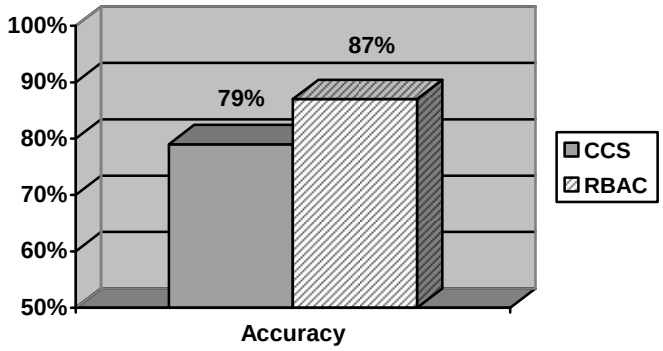


Figure 68. The accuracy of the subjects' access control setups.

This result tallies with what is to be expected. Due to its limitations there are situations in which the CCS mechanism cannot represent the subjects' preferences completely accurately. With three pieces of context and five potential recipients some inaccuracies are to be anticipated. The RBAC mechanism, on the other hand, can with the availability of 5 roles be completely accurately set up for the given scenario. Thus, it is anticipated that the RBAC mechanism should be the more accurate of the two, which the survey confirms.

Another finding, that is perhaps more interesting, concerns the subjects' perception of their ability to set up access control mechanisms. When asked how accurately they felt they had been able to set up the CCS mechanism, 65%

of the respondents stated that they had been able to express their preferences either very accurately or accurately. The respective figure for the RBAC mechanism was only 48%. Hence, the subjects in the sample felt that they had been able to express their privacy preferences more accurately with the CCS mechanism than with the RBAC mechanism, when in fact the opposite was true. This finding is illustrated in Figure 69. The CCS mechanism was also reported to have felt easier to set up by the subjects.

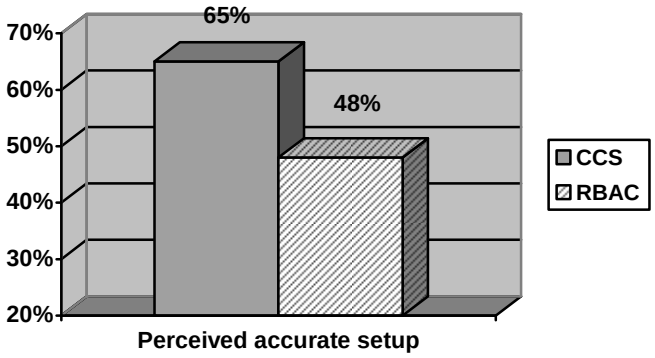


Figure 69. The subjects' perception of the access control setups.

This result indicates that it can be difficult for people to see the real effect of an access control mechanism at any particular state. It also suggests that they may not always be aware of how well they have managed to set up their privacy protection mechanism. Furthermore, the fact that the mechanism found to be the easiest to work with was also perceived to be the more accurately set up, seems to suggest that the ease of use may be misleading.

Finally, another finding concerns the cases where the subjects in the sample failed to set up the access control mechanisms to represent their preferences. With the CCS mechanism it was found that 70% of the inaccuracies lead to too high an access. The corresponding figure for the RBAC mechanism was 68%. Thus, in the cases were the access control mechanisms were inaccurately set up, it was found that this most often resulted in too high an access being granted. This finding is illustrated in Figure 70.

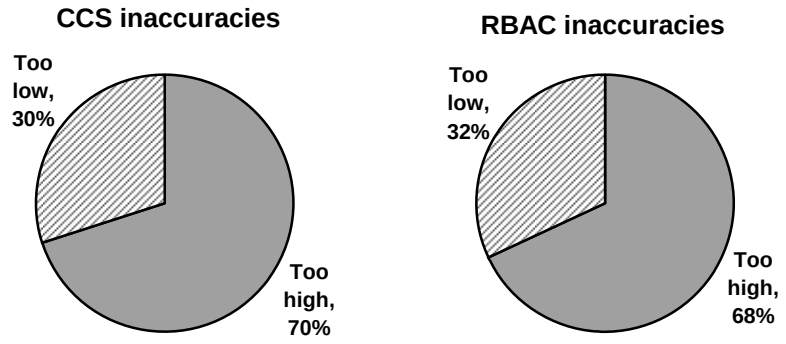


Figure 70. The effect of inaccurate representation of privacy preferences

This result is most worrying, especially since the other findings suggest that people may not always be aware of how well they have managed to express their privacy preferences. The overall consequence is likely to be a false sense of protection. This may subsequently affect people's trust in the technology, if and when they become aware that their expectations have not been met.

6.1.2.3. Validity to population

The validity of the survey results to the population is arguable given that the sample size was small. Whilst 31 subjects are considered to be enough to be able to draw some conclusions from the survey, in particular when the differences are clear, a larger sample is required to statistically demonstrate that the result applies to the general population.

The sample was also biased. With 70% of the respondents being male this gender was overrepresented in the sample. If this has affected the outcome of the survey is unclear, though given that the overrepresentation is relatively small it can be assumed that the effect also will be small. Also computer and internet literate users were clearly overrepresented. This is an outcome most likely resulting from the survey being performed online, and perhaps also due to its topic. This overrepresentation cannot be disregarded.

There is, however, evidence to suggest that even with a larger size the sample may still have been biased. For example, the 10th WWW user survey performed by the GVU Center at the Georgia Institute of Technology [GVU's WWW Surveying Team 1998] shows a similar bias despite having a much larger sample size. Furthermore, it can be argued that bias towards computer and Internet literate people is not a disadvantage. It is expected that the users of

ubiquitous computing systems are both computer and Internet literate. The number of households with internet access also increases steadily. Between 1998 and 2004 the proportion of UK households with internet access has gone from less than 10% to 52% [National Statistics 2005]. This of course will contribute to improve the population's computer and Internet literacy.

It is therefore believed that the sample provides a reasonable approximation of the population for the purpose of this survey.

6.1.2.4. Limitations of results

There are concerns with respect to how well the survey captures the everyday use of the access control mechanisms. Firstly, no training was given on how to use the mechanisms. Only a short description was provided as guidance. Secondly, the survey did not include any feedback of the outcome when setting up the access control mechanisms. In contrast when using the infrastructure, the administration console allows the user to view the actual outcome of the configuration and to fine-tune the access, if necessary. Hence, what the survey actually captures is the first attempt at using the mechanisms. Whilst this provides an indication of the initial accuracy that can be expected, it does not show how well the mechanisms can perform. In practise the accuracy is generally expected to be better, particularly with experienced users, but what improvement can be achieved for the respective mechanism is unclear.

Also, the general applicability of the results for the two types of access control mechanisms has limitations. The survey specifically compared the CCS and RBAC mechanism presented in this work (See above, section 4.5.2). Associated with these mechanisms are simplifications. Firstly, in CCS the classifications are associated with the identifiers of context information, not the data. There is therefore an element of uncertainty when performing the classification. It also implies that once released information is no longer associated with a classification. Secondly, the RBAC mechanism employs automatic role invocation. In practice this means that a user always receives the best possible access given their current set of roles. Hence, the exact roles assigned are in effect not important provided the overall access is correct. These simplifications can affect the reasoning and process of assigning access as well as affecting the

results directly. For example, with the RBAC mechanism the fact that access is achieved by invoking the wrong role is hidden, in some situations improving the measured accuracy and always increasing risks from impersonation. Hence, when comparing the results with other mechanisms these simplifications must be taken into account.

6.2. Development support

It has been suggested that an infrastructure, in addition to providing privacy protection, can make the development of context-aware applications easier and more rapid. This hypothesis has been based on the assumption that an infrastructure will reduce the complexity of the application design and that common functions can be performed by the infrastructure.

The approach taken to establish whether the infrastructure in fact aids the development of applications has consisted of a qualitative evaluation in three areas. Firstly, the usefulness of the feature set provided by the infrastructure has been evaluated. Secondly, the reduction to the code base of context-aware applications that can be achieved by using the infrastructure has been estimated. Finally, a measurement of the cost in terms of performance of using the infrastructure has also been made.

6.2.1. Feature set

To aid the development of context-aware applications the infrastructure has been developed to handle a number of key features. These features are now discussed from a developer's perspective. The discussion draws upon the experience and necessities of the agents developed for the infrastructure (See above, section 5.6) as well as the infrastructure itself (See above, sections 5.3 & 5.5). For each feature an analysis is made about the usefulness and the benefits it brings.

6.2.1.1. Privacy protection

The infrastructure's ability to protect the user's privacy has already been evaluated (See above, section 6.1). However, the inclusion of privacy protection in the infrastructure also affects the application developers and needs to be examined from this perspective as well.

Including privacy protection in context-aware systems can be a time consuming and costly process. The development of the privacy protection for the infrastructure has demonstrated this. From the requirements it can be seen that there are several areas that need to be covered including authentication, access control, anonymity, etc. In all of these areas various mechanisms must be chosen and implemented with care. Furthermore, from a developer's point of view the time spent on privacy protection can appear to add little direct value to a system. Adding additional functions has a more noticeable effect, even though this may leave the system open for abuse.

Given this, incorporating the privacy protection into an infrastructure is therefore seen to be beneficial. It removes the need for system and application developers to delve deep into the area. Instead the infrastructure provides a solution that takes care of the privacy protection and that is ready out of the box. And even though developers need to be careful not to void the protection, the factors they need to consider are minimised. This should save them time, time that then can be spent developing other parts of the system. Another useful aspect is that the responsibility for the implementation of the privacy protection is shifted away from each individual system or application. This minimises code duplication and thus also the risk of errors. Furthermore, if errors are found they can be corrected without having to update each individual system.

The web presence application takes advantage of the benefits provided with the infrastructure based privacy protection. The application, which publishes an entity's context on the web, has been developed without any access control mechanism itself. It simply publishes the information it can access from the entity's context manager. This of course reduces the development time significantly. It also makes the application much more flexible and responsive to changes in privacy preferences. Any changes made to the privacy preferences held by the context manager in the infrastructure, will be reflected by the web presence application. Hence, the process of configuring what is published or not is that of granting or revoking access to the information.

Relying on the privacy protection solution provided by an infrastructure has its disadvantages as well. A developer has no direct control over the privacy

protection mechanism. This includes the type of mechanism, its implementation, and how it is set up. This can be a problem under certain situations. For example, if it is desirable to run another access control system in parallel it may prove difficult to synchronise the systems. There may also be situations where the provided privacy protection is not appropriate. Assigning access per context item may for example be unnecessary.

6.2.1.2. Context storage

Another key feature of the infrastructure is context storage. The context manager in the infrastructure sits between the context producers and context consumers and manages the flow of information. This includes both temporary and long term storage of information (See above, section 5.3.3.1).

It is common for context-aware systems to require contextual information to be retained for later use. Independent of the length of time the storage of information is required it adds an overhead to the context-aware system both in terms of development and operation. It is particularly troublesome if individual applications are to store information. Furthermore, if the storage of a subject's contextual information is not coordinated this will not only lead to discrepancies but it will also make it difficult to control the flow of information.

Letting the infrastructure handle the storage of contextual information therefore has its benefits. From a developers perspective it enables context producers and context consumers to be developed without having to worry about how information is stored. It also widens the choice of devices that can be used when developing sensors and applications as the storage of information is shifted onto the infrastructure. In a ubiquitous environment where resources are scarce this can be very useful. From a wider perspective a notable benefit is that data duplication can be minimised and thus also potential discrepancies in the data. Looking at the implementation, the driver approach provides a flexible solution. By exchanging the data storage driver used by the context manager a system developer can customise how information is stored in the infrastructure. This is useful as the environments for which ubiquitous systems are developed vary along with the resources they offer. The developers can also implement custom

storage drivers and thus take control over how information is stored in the infrastructure.

All of the applications developed have benefited from the infrastructure's ability to store contextual information. In fact, none of the applications developed implement any mechanism for storing contextual information persistently. This has of course simplified their development. It has also the benefit of making breakdowns or theft less critical from an information perspective. For example, if the context-aware desk display were to break down no information would be lost as it is stored in the infrastructure by a context manager.

Placing the storage of contextual information in the infrastructure, however, comes at a cost. Even though it is possible to control how information is stored through the development of custom data storage drivers, the developer loses control over the storage process as a whole. It is only that last step, i.e. how information is physically stored, that can be influenced and even this is steered by the interface towards the context manager. Another disadvantage is that the storage of contextual information in the infrastructure is less efficient in terms of performance. Adding an intermediary infrastructure component, the context manager, between the context producer and context consumer removes the ability to subscribe to real-time streams of contextual information. The communication required to store and retrieve information also adds an overhead. Furthermore, in situations where connectivity is unreliable it becomes necessary to either cache information in the application or to utilise a local context manager. The former alternative forces developers to handle temporary storage of information anyway while the latter uses scarce resources.

6.2.1.3. Communication

Communication is a critical feature in context-aware systems. In these systems contextual information is generally captured, processed, and utilised by different components. Communication is the thread the ties them together.

There are three key aspects of communication that a developer needs to consider when developing a context-aware system. Firstly, how is information represented so that it can be conveyed? For this they need to design a data

format and a scheme governing how information is represented using the format. Secondly, how is the information conveyed? A mechanism needs to be developed with which information can be transported between components. Thirdly, how can the flow of information be secured? The developer needs to carefully select a solution that provides adequate security for the communication. Addressing these key aspects is time consuming for a developer. It is also unnecessary for each system to develop custom solutions as the problem is seldom unique. Furthermore, to achieve interoperability between different systems the solutions must be standardised.

With respect to communication the infrastructure provides developers with several benefits. To a large extent it removes the need for developers to consider communication. The infrastructure includes a predefined and extendable data format, saving developers design time. Embedded in the infrastructure is also a plug-in based communications framework that is ready to be used by developers. Thus developers benefit from not having to implement the communication. Another useful feature of the framework is that the details of the communication are transparent to the applications. This enables the developers to implement their system independent of how messages are transported or secured. In situations where a developer requires control over the communication this is also possible because of the framework's use of plug-ins. Hence, a developer is able to customise how information is both transported and secured.

The development of the iButton context capture application (See above, section 5.6.4) has demonstrated the benefit of the infrastructure's communication framework. Although the application is directly dependant on communication to function, i.e. to update an entity's context manager with any information captured, it was developed without implementing any communication mechanism. Instead the communication framework provided with the development toolkit was used to handle the communication. This shortened the development time. It has also made it possible for users of the application to choose what transport and cryptographic plug-ins to use.

The downside of using the infrastructure and its communication framework is that the developer becomes locked to a specific data format. Whilst this may not be a problem for the development itself, the performance of a finished system can suffer. For example, the XML based data format is not suitable for the transportation of very large messages. For each message a model is created in memory for further processing within the infrastructure. Large quantities of data are better sent using other means. Furthermore, the framework restricts a developer's ability to control the communication. Even though custom plug-ins can be developed and used, they need to comply with the guidelines of the framework.

6.2.1.4. Synchronisation

Synchronisation is another key feature in the infrastructure. Although it is a small part of the implementation it fulfils an important need. Furthermore, synchronisation between different systems can be difficult for developers to implement due to a lack of cooperation.

In a distributed ubiquitous system discrepancies and data duplication are often unavoidable. The lack of reliable communication between systems makes it necessary for information to be available locally to ensure uninterrupted operation. To recover eventual differences that come about during periods of offline operation it is necessary to synchronise the systems. In an environment filled with ubiquitous systems it will however not be practical for individual systems to handle the synchronisation of data. Implementing synchronisation routines for each system is a time consuming process. It also requires developers to know how to interface with each related system, thus placing unnecessarily high demands on them.

The implemented infrastructure provides built-in support for synchronisation. This moves the responsibility for keeping information synchronised away from individual systems and their developers. Doing so has its benefits. It removes the need to implement synchronisation routines in the individual systems developed, thus reducing their development time. It also provides developers with one issue less to concentrate their efforts on and in this way lowers the demands placed upon them. The implementation also has another useful feature.

Although the infrastructure handles the synchronisation of information, it is possible for individual systems to initiate the process. This feature is particularly valuable in situations where it is critical that information is up to date and synchronised.

In the deployed infrastructure context managers have been run both on mobile devices and on permanently online servers. The built-in synchronisation feature has been used to keep an entity's context managers synchronised. This allows the systems developed for the infrastructure to be used even when there is no connectivity back to the main server. Even in situations where the context changes with the state of the connectivity, e.g. location and availability of hotspots, it is useful to be able to update the contextual information on the move as this history can be kept. A key benefit is of course that the applications do not need to be adapted in any way to support the synchronisation of information.

However, implementing the synchronisation routines in the infrastructure rather than in individual systems is not only beneficial. A disadvantage is that developers lose control over the synchronisation process if they rely on the support provided by the infrastructure. This can be problematic as the ability to customise the process is limited. For example, only a very basic control is provided of the type of information being synchronised, data or context.

6.2.2. Code reduction

One way of making the development of context-aware systems easier and also more rapid is by reducing the amount of coding developers need to do. A reduction of the code base will also make maintenance less demanding, which is an added benefit. To estimate the extent to which the infrastructure allows coding to be reduced a scenario has been investigated.

6.2.2.1. Measure

To give a quantitative measurement of the amount of code required to implement a system is difficult. Several factors affect this including the design, choice of platform, coding style, programmer experience, etc. Hence, quantitative measurements of code use will be biased. This is not desirable. A coarser estimate of the amount of coding required has therefore been employed

in this evaluation. The approach taken has been to use the number of features required to be implemented as the indicator.

6.2.2.2. Scenario

Assume that a system is going to be built to make use of a network of location sensors deployed throughout a building. The system should initially support a simple application that allows a user location to be queried. But future plans are to make wider use of the location information collected as well as other pieces of context. To not discourage usage of the system it is required that individual users can determine for themselves who can query their location information.

6.2.2.3. Required features

A system for the outlined scenario requires a number of features to be implemented. Firstly, the system must be able to capture contextual information. For this a driver that can interface with the sensor network is required. This driver would then capture the raw location data picked up by the sensor network. Secondly, the system should be able to store, or at least cache, the captured location information so that it can be queried. Hence, the system needs to incorporate data storage. Thirdly, the system must contain mechanisms for privacy protection and security. At the very minimum authentication and access control is required. Hence, there need to be a information management component in the system. Fourthly, the system need to be able the handle location requests. A request handler component is thus required. The request handler would then interact with the information management component to fulfil requests. Finally, it is necessary to provide a user interface to the system. The user interface must allow users to express their privacy preferences and to post requests for location information. The interface could take the shape of a standalone application or perhaps be available over the web.

6.2.2.4. Implementations

How then does an independent implementation compare with one that uses the infrastructure?

Developing an independent system for the outlined scenario will require all of the features listed to be incorporated. Hence, without any external support the

implementation must contain code for context capture, data storage, privacy protections, security, request fulfilment, and user interaction.

A system developed for the same scenario, but that uses the support provided by the infrastructure to the full extent, instead only needs to implement two of the five required features, namely the context capture and the user interaction. Furthermore, the user interaction can also be simplified if the administrative console (See above, section 5.6.1) is used in conjunction with the system being developed as this would remove the need to provide an interface for the management of privacy preferences. To utilise the features of the infrastructure little additional code is required. Only a few lines of code are necessary to send and retrieve context to and from context managers, as illustrated in Figure 71.

```
//Load communication properties
Properties props = PropertiesReaderCcpp.readProperties(fileInputStream);

//Create and initialise communication handler.
ComHandler comHandler = new ComHandler();
comHandler.init(props);

//Create message
Message msg = new Message();
msg.setData(component,attribute-key,attribute-value);
...

//Post message
Message reply = comHandler.post(username, password, address, msg);

//Handle response
...
```

Figure 71. Using the infrastructure

Thus, the use of the infrastructure minimises the number of features a developer needs to implement. Whilst no measure has been made of the reduction in terms of lines of code, the experience from the implementation of the features in the infrastructure indicates that the reduction is significant. For example, the unoptimised implementation of the role based access control mechanism in the infrastructure, excluding P3P support, alone consists of more than 2300 lines (80 kb) of commented source code.

As a reference the iButton context capture application, including the graphical user interface, only consists of approximately 1200 lines (40 kb) of commented

source code. For the web presence application the source code consists of 450 lines (15 kb).

6.2.3. Infrastructure performance

Performance is an important factor when determining whether it is practical or not to use the developed infrastructure. The performance of the infrastructure has therefore been evaluated. In the evaluation the round-trip times and the cryptographic performance has been measured and analysed.

6.2.3.1. Platforms

The performance test has been run on two different devices, a handheld computer and a laptop.

The handheld device was an iPAQ 4150 PDA. It is equipped with an Intel PXA 255 processor, 400mhz ARM X-Scale, and 64 Mb of RAM whereof 15 Mb was used for storage. Also available was additional storage in the form of a Secure Digital flash card with a stated maximum transfer rate of 22 Mbit/s. The infrastructure components were run from the flash memory and also utilised it for storage. The operating system on the device was Windows Mobile 2003. The Java virtual machine used was IBM's J9 with the CDC configuration and the Personal Basis Profile.

The laptop device was an Evo N1015. It is equipped with a Mobile AMD Athlon XP 2000+, 1.66 Ghz and 512 Mb of ram. The infrastructure was run from the hard disk and also utilised it for storage. The operating system on the device was Windows XP Professional SP1. The Java virtual machine used was Sun's J2SE 1.3.

6.2.3.2. Round-trip

One measure of performance is the round trip time, i.e. the time it takes for a request to be fulfilled. This includes the time it takes for a request to be delivered, processed, and then have a reply returned. The evaluation performed has measured the round-trip time for both context-producers and context consumers with two different sized payloads.

To measure the round-trip times a simple test agent has been developed that first takes the role of a context producer and then of a context consumer. The agent will therefore write and read to a context manager depending on its current role. To measure the round-trip time a timestamp is taken directly before a request is sent and immediately after the response has been received. The difference between those timestamps is the round-trip time. To avoid any bias the payload written and read by the agent is pseudorandom and stays the same throughout the evaluation.

The performance test was carried out in an isolated infrastructure set up dedicated to the test. The setup consisted of a single context manager and a test agent, both running locally on the device. The components were run in separate Java virtual machine instances and they communicated over a socket connection. The payload sizes used during the test were 1024-bytes and 10240-bytes. Added to this was the overhead from the message format. For example, a request to write 1 Kb of data results in a message size of 2295-bytes, the corresponding value for 10 Kb is 14935-bytes. During the test, each operation was run 10 times from which an average was then calculated. Figure 72 contains the results.

	1024-bytes	**10240-bytes**
iPAQ 4150	1785 / 784	46182 / 22488
Evo N1015v	149 / 90	6643 / 2851

Figure 72. Write/read time in ms

The results show that it takes approximately 1.8 seconds to write 1 Kb of data to the context manager on the PDA and approximately 0.8 seconds to read it back. For many purposes this is considered an adequate level of performance, particularly if context updates are infrequent. However, it is not sufficient for the infrastructure to serve applications that require context updates in real-time. Hence, in practise the application domain that can benefit from the infrastructure will be limited. The figures for the laptop, which are 10 times faster however, show that as processing power and memory increase the performance will improve significantly.

The test also shows the write operation on both types of devices to be significantly slower than read operation. This is to be expected with the current implementation. The data storage caches the available information in memory. Hence, data can be read quickly. The data storage will also commit data to disk after each write operation. Hence, the write operations are expected to take longer. What was unexpected though, was the detrimental effect the increased payload had. Increasing the payload 10 times, to 10 Kb, causes the round-trip to take 30 times longer. This is a significant increase that indicates that the infrastructure in its current state does not scale to handle large payloads. Hence, for optimal performance the infrastructure is best used with smaller payloads. To what extent careful optimisation can improve the situation is still unclear.

6.2.3.3. Cryptography

Given the resource constraints in ubiquitous computing systems it is important to establish to what degree cryptography affects performance. This will help to determine the right balance between performance and secrecy of communication. The evaluation performed has measured the cryptographic performance of the implemented RSAAES mechanism (See above, section 5.3.1.3) with respect to key generation and the encryption/decryption of data.

To test the RSAAES performance a stand-alone test application was developed that utilises the cryptographic library in the infrastructure. The test application measures the key generation time by taking a timestamp immediately before and after a key is generated. The difference between those timestamps is taken to be the key generation time. The same approach is taken to measure the time it takes to encrypt and decrypt data. To avoid any bias both the data and the RSA key were pseudorandom and constant throughout the evaluation.

The RSAAES tests were performed on the laptop and the handheld device previously described (See above, section 6.2.3.1), which were temporarily dedicated to this task. The RSA key sizes used during the test were 512-bits and 1024-bits. The AES key size was kept constant at 256-bits. As with the round-trip performance test the two data sizes 1 Kb and 10 Kb were used. Each operation was also run 10 times from which an average value was calculated. The results are presented in Figure 73 to Figure 75.

	512-bits	1024-bits
iPAQ 4150	9532	128916
Evo N1015v	80	1191

Figure 73. RSA Key generation time in ms

The RSA key generation was found to be the most time consuming operation in the test. This was expected as a previous benchmark [Osbakk, Ryan 2004B] has shown this to be the case. What was unexpected though was the length of time it took to generate keys on the PDA. The generation of the 1024-bit keys, the now recommended minimum length by RSA Laboratories [Kaliski 2003], took on average approximately 129 seconds. This is significantly longer than the approximately 4 seconds found in the previous benchmark performed on the same device [Osbakk, Ryan 2004B]. Since the previous benchmark both the RSAAES implementation and the cryptographic library used [Bouncy Castle 2003] have remained unchanged. The only difference between the set ups is the Java virtual machines used. This test used IBM's J9 whilst Insignia's Jeode was used before. Hence, this indicates that the choice of JVM can make a significant difference to the cryptographic performance. The result also further emphasises the importance of reusing the RSA key pair as it would not be feasible to regenerate the key pair for every request or even every new session. A one off delay of 129 seconds for the key generation, although clearly undesirable, is still considered to be acceptable. It does suggest though that persistent keys must be generated in advance rather than upon the first request as done in the current infrastructure implementation.

	512-bits	1024-bits
iPAQ 4150	157 / 529	178 / 920
Evo N1015v	4 / 14	4 / 36

Figure 74. RSAAES encryption/decryption time in ms (1024 bytes)

The performance test measured the time to encrypt and decrypt 1 Kb of data to be on average approximately 0.18 and 0.92 seconds, respectively, on the PDA. These operations are also slower in comparison to the benchmarks run on the Jeode JVM, though not to the same degree [Osbakk, Ryan 2004B]. This slow down makes cryptography more of an issue than previously believed. For

example, the encryption of a 1 Kb read request takes almost as long as the round-trip. However, the fact that the operations can be performed significantly faster on another JVM leaves room for optimisation. It should also be noted that on the laptop device the measured times are so short they are largely insignificant. Hence, as the processing power increases on ubiquitous devices cryptography is likely to become less of an issue. This view is supported by the results from the previous benchmark as it showed the cryptographic performance to improve approximately by a factor of 2 between an iPAQ 3660 and an iPAQ 4150 [Osbakk, Ryan 2004B].

	512-bits	**1024-bits**
iPAQ 4150	761 / 1235	749 / 1620
Evo N1015v	30 / 30	15 / 46

Figure 75. RSAAES encryption/decryption time in ms (10240 bytes)

Finally, the test also shows the encryption and decryption times to be increased with larger pieces of data, as expected. What is important to note, though, is that in contrast to the round-trip times the cryptographic performance increase at a slower rate than the data size. Increasing the data size by 10 times resulted in an increase in the order of magnitude of approximately 4 and 2 times for encryption and decryption respectively on the PDA. This result is in line with expectations as the work performed by the slower asymmetric cipher, RSA, in the RSAAES combination is unaffected by the data length. An increased payload size will therefore effectively make the latency introduced by the RSAAES cryptographic mechanism less significant overall.

6.3. Comparison with related work

In general there is more than one way of doing things. Providing privacy protection and development support for context-aware systems is no exception. It is therefore necessary to also evaluate how the developed infrastructure compares with related work.

The evaluation consists of a qualitative comparison with the approaches taken in three related projects: the privacy awareness system, solar, and the EQUIP Platform [Greenhalgh 2002]. The aim is to establish if the approach taken in this

work can be motivated given the existence of alternative approaches. Hence, the evaluation has been made with respect to the background, aims, and requirements presented in Chapter 1 to Chapter 4.

6.3.1. Privacy-awareness system

The privacy-awareness system (pawS) [Langheinrich 2002], is a related research project also investigating a comprehensive solution for improving privacy in ubiquitous computing. Just as the infrastructure presented in this thesis, pawS, covers a wide range of issues including secure communication, anonymity, and access control. Furthermore, pawS also utilises the Platform for Privacy Preferences Project (P3P) as a supporting technology. However, the similarities end there. There are a number of important differences though compared to this work, each of which has its advantages and disadvantages.

Firstly, the foundation for the work on privacy in pawS is six design guidelines concerning notice, choice and consent, anonymity and pseudonymity, proximity and locality, adequate security, and access and recourse [Langheinrich 2001]. In contrast this work is based on a model of privacy with the focus on controlling the release and leakage of information. A key difference is therefore the width of the foundation, with pawS having a broader scope. The broader scope makes desirable requirements, e.g. security, more evident in pawS. This is certainly advantageous. However, from the point of view of this project it is not necessary as it is not a problem to specify requirements later.

Secondly, pawS emphasises the importance of explicitly making users, or their agents, aware of when data is being collected. This should be compared to the approach taken in this work where notice is only given during the agreement stage of the process of disclosure, which can occur anytime before the actual data collection. The advantage of the approach taken in pawS is that users can be made aware of potential data collection before it occurs, providing them with an opportunity to oppose the collection of data. Hence, the notice provides a mechanism enabling information leakages to be managed. An added benefit is that previously unknown services can be presented to a user. The disadvantage, however, is that this approach may provide a false sense of security when

systems do not follow the rules. For the purpose of this work leakage management is not required by the adopted privacy model.

Finally, the privacy-awareness system's architecture assumes the presence of Internet connectivity. This is necessary because a user's mobile privacy assistant offloads the decision making onto a personal privacy proxy situated on the Internet. In contrast the infrastructure developed in this work is not dependent on Internet connectivity for it to work. The infrastructure can run locally on a disconnected network or just on a device. The advantage of offloading demanding operations to a proxy on the Internet is that the mobile clients can conserve their resources. The disadvantage, however, is that the use of the system becomes limited to where there is connectivity. This hinders ubiquitous operation, which is essential for the infrastructure developed in this work.

6.3.2. Solar

The Solar system [Chen, Kotz 2002] [Minami, Kotz 2002], like this work, provides a privacy enhancing infrastructure that supports the development of context aware applications. Furthermore, Solar also focuses on access control as the means of improving privacy. Thus there are clear similarities between the projects. However, once again there are a number of important differences compared to this work, each of which has its advantages and disadvantages.

Firstly, the Solar system uses a subscription model to distribute contextual information whilst this work has employed a request-based model. The advantage of the subscription model approach is that contextual changes propagate to the listening applications in the system when they occur. It also removes the need for applications to poll for contextual information or to add their own subscriptions support. However, to manage the subscriptions a central authority, a star, is used. This centralisation, although possible in environments with reliable connectivity, does not allow the system to fully support ubiquitous operation where connectivity is limited and unreliable. A request-based approach, as employed in this work, is more suitable there.

Secondly, in Solar the processing of contextual information is integrated within the data flow. Operators are used to filter, transform, merge, and aggregate information between the source and the application. These operators can be distributed in the network and they interact by subscribing and publishing event streams. Furthermore by chaining operators a sequence of transformations can be performed. In contrast the approach taken in this work typically results in the processing being centralised to the context managers. Context producers deliver information directly to a subject's context manager. Information is then processed by the context manager itself or by services which return their output to the context manager again. The advantage of Solar's approach is that it allows complex operations to be distributed and performed by a graph of simpler operators. As transformations are performed by the operators in the system it ought to also support reuse of transformation sequences better. The disadvantage of Solar's approach, though, is that the flow of information cannot be controlled between processing steps. Thus in Solar access to information is first enforced at the root of a subscription tree, as the last step before passing on an event to an application. This is not compliant with the requirements of this work. A subject must be able to control the flow of information about them.

Thirdly, a difference exists not only in how access is enforced but also in how access controls are used. As previously described (See above, section 2.6.4) each event in the Solar system is tagged with an access control list (ACL) upon creation. The ACL is then modified appropriately if and when the event is transformed by operators. Hence, the ACL associated with an event published by an operator is derived from the ACLs of the events forming the input. In addition to deriving access, operators can restrict the access further and users can ease the control. This should be compared to the access control mechanism in this work that requires users to explicitly set permissions for each context item. The advantage of Solar's approach is that it scales better. As the number of context items sensed in the environment increases it becomes increasingly difficult to individually control access to them. Thus deriving access control settings from those initially set by the source is beneficial. Once again though, Solar's approach provides the users with less than full control over the flow of

information. Although users are able to affect how the access control is derived at each operator, their customisation is limited to attaching their own relaxation functions. The consequence of the assumptions made in this work is that full control is required.

Finally, in Solar neither authentication nor secrecy of communication is directly addressed in conjunction with privacy. These issues are instead assumed to be provided by the system as a whole. In contrast the approach taken in this work has been to address authentication, access control, and secrecy of communications together. Whilst focusing on access control separately may allow more progress to be made in this specific area, it is believed to be more beneficial for a system as a whole to provide a more encompassing solution.

6.3.3. EQUIP

The EQUIP software platform [Greenhalgh 2002], developed within the EQUATOR Interdiciplinary Research Collaboration, is another project that has opted for an infrastructure approach. Whilst there are some basic similarities to the infrastructure presented in this work, the design of the EQUIP platform differs significantly.

Firstly, EQUIP uses the notion of dataspaces to hold shared information. In contrast this work uses context managers to hold information about their associated entity only. Furthermore, in EQUIP any application is allowed to create a dataspace server and share information whilst in this work the information sharing is separated from the applications. The advantage of using dataspaces is that it allows multiple applications to work on the same data. This allows for better collaboration, less redundancy, improved scalability, and more independent services. The disadvantage is that the ownership of information is not clear, which raises the question of who should control access. Given the requirements of this work the use of dataspaces with shared information is therefore considered inappropriate.

Secondly, the EQUIP platform employs an event-based model with state sharing to distribute information whilst this work implements a request-based model. The advantage of an event based mechanism is similar to that found with the

subscription model used by Solar (See above, section 6.3.2). It allows changes to information to propagate to listening applications, removing their need to constantly poll for updates or add custom subscription support. Furthermore, in contrast to Solar, EQUIP does not rely on a central authority for managing subscriptions but associates them with the dataspace. Operation in ubiquitous computing environments with potentially unreliable communication is therefore possible. A disadvantage of the event-based model can be argued to be the overhead required to retrieve information, though this is only valid in situations where updates are not required. It must therefore be concluded that an event-based model, like that used by EQUIP, is advantageous. However, from the point of view of this project a request-based model is sufficient to fulfil the requirements.

Thirdly, EQUIP supports distributed computation natively. The items being shared in the dataspaces are software objects capable of both holding data and providing methods for computation. In comparison the messages sent in the infrastructure presented in this work are only bearers of information. Distributed computation is achieved through the use of agents, proxies, and resource extension plug-ins. The advantage of embedding methods for computation directly in the data items is that this functionality is available to all applications using the dataspace without additional configuration. The disadvantage is that limitations are placed on what devices can be used with the platform. The devices must support a runtime environment compatible with the data items. Furthermore, it may not be possible for the producer of a data item to foresee all types of distributed computations required by applications. Which approach is the most suitable therefore depends on how the infrastructure is intended to be used. In this work an emphasis is placed on maximising device compatibility.

Finally, another important difference between EQUIP and the infrastructure presented in this work is how privacy and security issues are handled. The former provides limited security and no privacy protection whilst the latter addresses both areas. What is available in EQUIP is the ability to protect access to the dataspace using a shared secret (password); no further access control is

provided[1]. The advantage of not providing a more extensive mechanism for privacy and security is that the complexity of the platform is reduced. Also performance should, in theory, be better as less processing is required. The disadvantage is that subjects are not able to control the flow of information about them. Furthermore, the platform becomes dependent on an appropriately configured network for security. The level of protection available in EQUIP is therefore not sufficient to meet the requirements of this work.

It should be noted that on the whole EQUIP focuses on a different, but related, problem; that of sharing information between ubiquitous computing devices and virtual environments. Thus, the existence of fundamental differences is to be expected.

6.4. Summary

In this chapter the developed infrastructure has been evaluated. In particular the evaluation has examined the privacy protection, the development support, and how the infrastructure compares with related work.

A qualitative examination of how well the privacy requirements have been fulfilled revealed an overall positive result with all requirements being considered by the infrastructure design. There are limitations though to the solutions provided and the circumstances under which they work. In particular further work in the areas of anonymity/pseudonymity and security would be beneficial. Interesting results were also found with the user study evaluating the performance of the candidate access control mechanisms. Although the role based access control model implemented in the infrastructure was found to be the more accurately set up, as expected, the study showed a mismatch between the actual and perceived accuracy. This indicates a difficulty in seeing the real effect of an access control mechanism.

To evaluate the development support a qualitative evaluation of the features set was performed where the implemented agents served as case studies. The investigation concluded that the features included in the infrastructure are

[1] Confirmed by C. Greenhalgh in an email communication on 26 April 2007.

indeed useful and will benefit developers. However, it was also recognised that each feature imposes distinct limitations on the system being developed. Hence, the ease of development comes at cost. An estimate of the reduction in code the use of the infrastructure can bring was also made by examining a hypothetical scenario. It showed that a significant reduction is possible. Furthermore, to establish whether the use of the infrastructure is feasible its performance was measured. The results revealed that for small messages (~1 Kb) decent performance is attained, with read and write operation taking less than 1 and 2 seconds respectively on a mobile device. Also the use of RSAAES cryptography was found to be feasible. The infrastructure is however not suitable for use in real-time systems or with large messages.

To establish whether the approach taken in this work can be motivated, given the existence of other approaches, a qualitative comparison was conducted with three related projects: the privacy awareness system, solar, and EQUIP. The comparison was made with respect to the background, aims, and requirements of this project. In the evaluation several differences were identified, each with there own advantages and disadvantages. In the context of this work however, it has been concluded that the approach taken can be motivated even if there are areas that can be improved upon.

CHAPTER 7

CONCLUSION AND FURTHER WORK

Privacy is a matter of utmost importance and surveys have shown it to be a real concern in our information society. Ubiquitous computing and context-awareness further heighten these concerns. The work presented in this thesis has taken some steps towards improving the situation through the development of a privacy enhancing infrastructure.

This chapter concludes this thesis. It begins by providing a summary of the work presented. After this some areas of further work are discussed that originate from the limitations of the project and the results found. Finally, the initial goals are revisited and an overall conclusion presented.

7.1. Summary

In Chapter 1 the background for the work in this thesis was presented. This included work in the fields of ubiquitous computing, context-awareness, and privacy. From the background it has been concluded that the early work on ubiquitous computing still largely guides the research in the field. Context-awareness was shown to be an integral part of ubiquitous computing, allowing enhanced services to be provided. Privacy was established as a long standing desire made more acute with the introduction of ubiquitous and context-aware computing. Recent work concerning privacy on the web has helped to bring the issue to life. Thus, the need to combine the work of the three fields was found.

In Chapter 2 the project that forms the basis of this thesis was outlined. The need for improved privacy protection and development support for context-aware systems was substantiated and described as the motivation for the project. The research was fixed to focus on applied research performed iteratively. An

infrastructure approach was established to be advantageous for the work. Results from related work were also examined and taken onboard.

In Chapter 3 two conceptual models were presented, one with respect to context and another to privacy. Context was defined in terms of the existence of relationships between entities and data values. Arbitrarily complex networks of relationships were demonstrated to exist. Privacy was defined in terms of information flow and control. It was assumed that the ideal level of privacy in online environments is equal to that experienced offline. Furthermore, it was demonstrated that a subject's control is limited to controlling their own disclosures, presence, actions, and receptiveness. The use of the models was also clarified using a scenario.

In Chapter 4 the privacy enhancing infrastructure was introduced. Requirements were extracted from the project's aim, the background of the work, and the conceptual models. The need to prioritise privacy and to use a modular design was shown. It was also established to be necessary to restrict the depth of the context model and to target a subset of devices. In order to uphold these preconditions a decentralised infrastructure was presented. Furthermore, alternatives for the privacy protection were discussed and the issue of context communication addressed.

In Chapter 5 the implementation of the privacy enhancing infrastructure was presented. The target platform and the advantages of a platform independent implementation were described. How the context manager handles the storage and protection of contextual information was demonstrated, including its use of exchangeable modules. It was also shown how the use of catalogue services and proxies provides the means to simplify the addressing of entities, translate between protocols, and provide anonymity. Furthermore, a number of agents utilising the infrastructure to store and retrieve contextual information were presented.

In Chapter 6 the infrastructure developed was evaluated. A qualitative requirements fulfilment analysis showed the privacy requirements to be met with the design of the infrastructure. From a user survey performed it was

concluded that the role based access control mechanism was the more accurate mechanism of the candidates presented in Chapter 4. Evidence was also found that indicate it is difficult for users to see the real effect of an access control mechanism. An analysis of the features in the infrastructure, from a developer's perspective, showed them to be useful. Examining a hypothetical scenario it was also demonstrated that a reduction of application code is possible when using the infrastructure. Furthermore, measuring the performance of the infrastructure showed it to be sufficient for the purposes of the agents developed, but not for applications demanding real-time context updates. Finally, with respect to the preconditions of this project it was found that the approach taken in this work can be justified in comparison with approaches taken in related work.

7.2. Contributions

The intentions of this work were to contribute in four areas relating to the development of privacy-friendly context-aware systems.

Firstly, the overall level of privacy in context-aware systems was to be improved. In this work it has been shown how a subject's privacy can be protected by providing means of controlling the disclosure of information. The privacy-enhancing infrastructure developed demonstrates the use of this approach in context-aware systems. The infrastructure also demonstrates the feasibility of using cryptography to ensure the confidentiality of information in transit even in ubiquitous computing environments.

Secondly, different access control mechanisms were to be evaluated. In this work a classification and clearance scheme and a role based access control mechanism have been evaluated, revealing the latter to be the more accurate both in theory and in practise. Worth noting though, is that the users perceived their performance with the classification and clearance scheme to be better. Furthermore, the work has also shown how the Platform for Privacy Preferences Project (P3P) [World Wide Web Consortium 2002A] can be used to support access control with respect to previously unknown users, though due to poor tools this mechanism proved impractical.

Thirdly, support for easy and rapid development of privacy-friendly applications was to be provided. This work has addressed this issue with the development of the privacy-enhancing infrastructure. It has been shown how the infrastructure minimises the development effort by providing common features required by context-aware applications. Furthermore, it has been demonstrated that it is feasible to utilise an infrastructure approach for general purpose applications even with resource constrained devices.

Finally, third-party infrastructures were to be protected. In this thesis it has been described how resource extension plug-ins can be used to integrate with other infrastructures, thus protecting access from the outside. A plug-in has also been developed that demonstrates the integration between the privacy enhancing infrastructure developed in this work and the MobiComp infrastructure [Ryan 2005].

7.3. Further work

There are many aspects to the development of a privacy enhancing infrastructure for context-awareness. The scope of this work, however, has implied that only a selection of the relevant aspects has been covered. Furthermore, the evaluation also highlights some issues that need to be addressed.

The key candidates for further work will now be discussed.

7.3.1. Access control

Access control plays a central role in providing privacy. In this work a classification and clearance scheme and a role based access control mechanism have been evaluated. From the experience gained, three aspects have been identified as candidates for further work.

Firstly, neither of the two access control mechanisms evaluated handle dynamic privacy preferences. This implies that it is not possible to describe privacy preferences that are context sensitive, a feature that is deemed desirable. An investigation, however, has shown promising initial results concerning an extension to the current role based access control addressing dynamic preferences using the concept of a privacy invasive value [Osbakk, Ryan

2004A]. Further work, examining the extension and how it performs with respect to related work on dynamic access control [Covington, Long et al. 2001] [Zhang, Parashar 2004], is believed to be warranted.

Secondly, from the user survey undertaken to evaluate the classification and clearance scheme and the role based access control mechanism, it was concluded that difficulties exist for people to see the real effect of an access control mechanism. When this causes too high an access being granted, as it was found to do in the majority of the cases examined, it leads to a false sense of protection. This is clearly undesirable and further work is necessary to attain an understandable, yet powerful, access control mechanism.

Thirdly, this work has not investigated how to represent the access controls in a standardised form. Instead solution specific alternatives have been used. Whilst this has worked well for the purposes of this research, the use of a standardised format would be desirable in cases of public deployment. Research by others has shown good results using X.509 attribute certificates with Role Based Access Control [Chadwick, Otenko et al. 2003][Chadwick, Otenko 2004], making it attractive an alternative. A candidate for further work is therefore to investigate the use of X.509 certificates, particularly in conjunction with a privacy invasive value approach. Another alternative to investigate is the use of a custom but publicly specified role definition language, as used by others [Mascone 2002].

7.3.2. User interaction

The privacy enhancing infrastructure developed focuses on the technical aspects involved when building privacy-friendly context-aware systems. User interaction has therefore not been prioritised. However, the discrepancy found in the undertaken survey between the actual and perceived performance in setting up the access control mechanism emphasises the need for informative interfaces. Furthermore, the lack of tools found even for standardised mechanisms such as P3P indicates that it is not possible to rely on external tools. An important piece of further work is therefore to increase the scope to include user interaction.

There are different directions future work on user interaction can take. Firstly, the use of traditional graphical user interfaces can be optimised to improve their performance. For example, the work others have performed on their user interfaces for privacy control highlights several issues [Lau, Etzioni et al. 1999] [Lederer, Hong et al. 2003] and has shown that significant improvements can be achieved [Brostoff, Sasse et al. 2005]. Secondly, research into the use of more novel interfaces for privacy control is called for. After all a key aspect of ubiquitous computing is to change the way we interact with computational devices, so should this not also include privacy control?

7.3.3. Trust management

In the context model presented, and subsequently the infrastructure, an important factor in determining whether or not to release information is the recipient's trustworthiness. Unfortunately, given the time constraints of this project it has not been feasible to study the formation and management of trust. Instead the work has relied on existing social mechanisms. Trust management is therefore a candidate for further work.

Within the area of trust management there are several aspects that are of interest to the further development of the infrastructure. Firstly, it would be useful to study the processes involved in the formation and management of trust in everyday offline situations. This information may then be used to evolve the conceptual model of privacy. Secondly, it would be interesting to evaluate the effect of using trust management tools in conjunction with the infrastructure's access control mechanism. Could they perhaps be used as an alternative to P3P when interacting with previously unknown participants? Finally, the current infrastructure does not provide any mechanism to discourage improper use of information once disclosed, but relies on existing social mechanisms. Research into the use of trust management as an enforcement mechanism in the infrastructure is therefore another candidate for further work.

7.3.4. Security

For privacy protection to be effective, a system must be secure. With the privacy enhancing infrastructure a basic level of security is provided. However, as discussed in the evaluation, the employed security mechanisms are limited

and have scalability issues. The improvement of security within the infrastructure is thus a candidate for further work.

Given the scalability issue with the current authentication mechanism, improving the authentication is a priority. In particular further work into the usage of a public key infrastructure is deemed important. With digital certificates entities could securely authenticate themselves in a uniform manner within the privacy enhancing infrastructure. It is recognised that intermittent connectivity and resource constrained devices, both common properties in ubiquitous computing, pose a problem. Connectivity is required to validate certificates and the signing/verification of digital signatures are computationally intensive. However, given that connectivity is also required to distribute context and that the communication already uses public keys to ensure confidentiality it is believed these issues can be addressed.

Further work securing the integrity of communication and the availability of the infrastructure is also desirable. However, for the purpose of the infrastructure these areas are not prioritised.

7.4. Conclusion

This thesis has argued that there is a need for privacy protection and development support for privacy-friendly context-aware systems. The hypothesis presented was that this could be provided with a privacy-enhancing infrastructure.

To investigate the validity of the hypothesis a privacy enhancing infrastructure for context-awareness has been developed and implemented. The infrastructure is decentralised and is capable of ubiquitous operation on handheld devices. It supports the development of applications by handling the storage and privacy protection of contextual information and by providing desirable features like synchronisation. To protect the privacy of its users the infrastructure provides authentication, access control, confidential communication, and support for anonymity.

The evaluation shows that the privacy enhancing infrastructure developed is capable of providing a reasonable level of privacy protection in line with the

requirements of the conceptual model of privacy. It also shows that the infrastructure provides significant support for the development of privacy-friendly applications and that it can be used with resource constrained devices. Hence, the hypothesis is concluded to be verified as true.

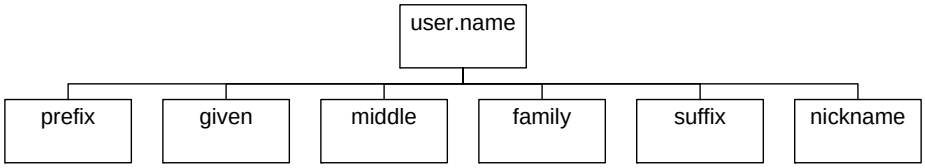
Nevertheless, significant limitations and issues have also been found in this investigation. Further work is therefore clearly desirable, particularly with respect to access control, user interaction, trust management, and security. It is hoped that the initial steps taken in this work will inspire further research into these aspects of privacy protection.

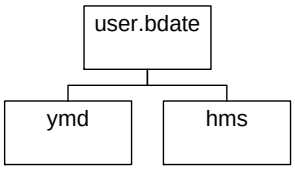
APPENDIX

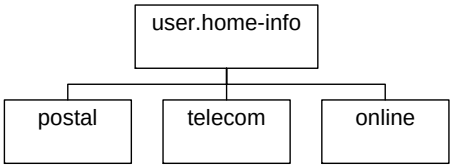
A Context vocabulary

A.1 Components

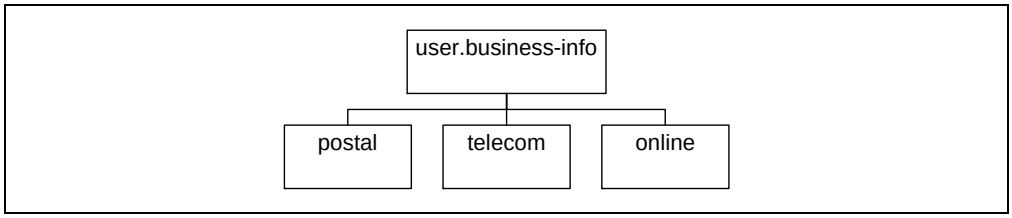
User

user.name	<string>
The user's name.	
	

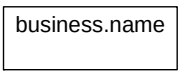
user.bdate	<string>
The user's birthdate.	
	

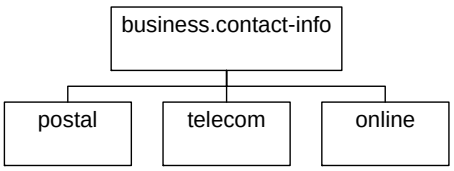
user.home-info	<string>
The user's private contact information.	
	

user.business-info	<string>
The user's professional contact information.	

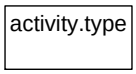


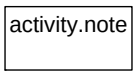
Business

business.name	<string>
The entity's name.	
	

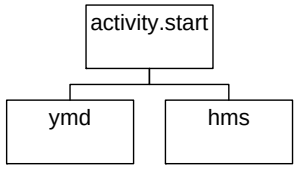
business.contact-info	<string>
The entity's contact information.	
	

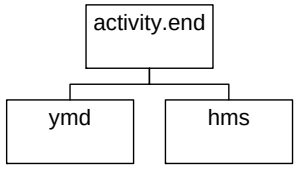
Activity

activity.type	<string>
The type of activity.	
	

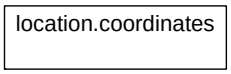
activity.note	<string>
Further description of the activity.	
	

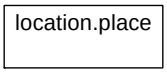
activity.start	<string>
When the activity commenced.	

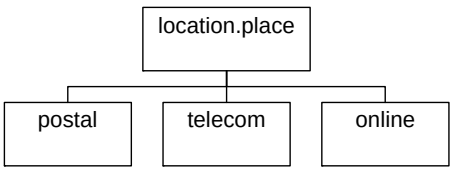


activity.end	<string>
When the activity expect to finish.	
	

Location

location.coordinates	<string>
The location expressed in coordinates.	
	

location.place	<string>
The location expressed as a place name.	
	

location.contact	<string>
The location specific contact details.	
	

A.2 Attributes

General

value	<string>

The component's value.

timestamp	<long>
The value's timestamp. (Specified as the difference in milliseconds between current time and midnight 1 January 1970 UTC).	

Content

content-type	<string>
The MIME type of the data.	

content-transfer-encoding	<string>
The encoding of the data.	

Quality

accuracy	<integer>
Estimate of the data's accuracy.	

confidence	<integer>
Confidence in estimated accuracy.	

Validity

lifetime	<long>
The estimated lifetime of data. (Specified as milliseconds after the timestamp).	

B Context-profile extension

B.1 Header attributes

General

status	<1-2>
The request's status.	
1) ok – request performed without errors. *2) error – errors occurred when performing the request.*	

command	<1-3>

The command to be performed.
1) availableData – lists the data available. *2) sycnhronise – performs a synchronisation.* *3) legacyCommand – performs a legacy command.*

Authentication

username	<string> | <1>
The requestor's identity.	
1) anonymous – request from an anonymous agent.	

password	<string>
Password authenticating the claimed identity.	

onBehalfOf	<string>
Indicates that the request is forwarded by a privileged user responsible for authenticating the requestor.	

view	<string>
Allows an administrator to assume the identity of a user.	

Confidentiality

publicKey	<string>
Holds public key in plug-in dependant binary format (Base64 encoded).	

cipher	<string>
The name of the cipher for which a public key is valid.	

encrypted	<string>
Indicates the presence of encrypted data with the name of the employed cipher.	

encryptedData	<string>
Holds encrypted data. (Base64 encoded).	

P3P

p3pId	<string>
The requestor's identifier as referred to in P3P rulesets.	

p3pUri	<string>
URI to the requestor's P3P policy.	

p3pPolicy	<string>
The requestor's P3P policy. (Base64 encoded)	

Synchronisation

reference	<string>
The URI to the remote Context Manager. (CM URI).	

syncMode	<1-3>
The mode of synchronisation.	
1) synchronise – update local and remote information (default). *2) updateLocal – update local information only.* *3) updateRemote – update remote information only.*	

syncType	<1-3>
The type of information to be synchronised.	
1) all – synchronise context and data (default). *2) context – synchronise only context.* *3) data – synchronise only data.*	

syncConflictResolution	<string>
The method by which conflicts are resolved.	
1) noChange – no change (default). *2) useLocal – use the local information.* *3) useRemote – use the remote information.*	

B.2 Body attributes

General

action	<1-3>

The action to be performed.
1) read – retrieves the context item (most recent). *2) write – stores the context item.* *3) history – retrieves a historic context item.*

status	<string>
The context item's status.	
1) ok – action performed without errors. *2) error – error occurred when performing the action.*	

Data list

current	<long>
The timestamp of the most recent occurrence of the context item. (Specified as the difference in milliseconds between current time and midnight 1 January 1970 UTC).	

historic	<long> | <bag>
A timestamp or bag of timestamps of the context item's historic occurences. (Specified as the difference in milliseconds between current time and midnight 1 January 1970 UTC).	

C Interface specification

C.1 Datastorage driver interface

net.osbakk.pi.cm.dataStorage
Interface DataStorageDriverInterface

public interface **DataStorageDriverInterface**
The DataStorageDriverInterface defines the methods the DataStorageDriver class must implement.

Method Summary	
void	**clear**() Clear datastorage.
void	**connect**() Activates the datastorage.
void	**deleteData**() Deletes data.

void	**deleteData**(java.lang.String category) Deletes data.
void	**deleteData**(java.lang.String category, java.lang.String key) Deletes data.
void	**deleteData**(java.lang.String category, java.lang.String key, java.lang.Long time) Deletes data.
void	**disconnect**() Inactivates the datastorage
void	**flush**() Flushes queue.
java.util.Vector	**getCategories**() Retrieves categories.
java.lang.Object	**getData**(java.lang.String category, java.lang.String key, java.lang.Long time) Retrieves data.
java.util.Vector	**getKeys**(java.lang.String category) Retrieves keys.
java.util.Hashtable	**getKeysTable**(java.lang.String category) Retrieves keys.
java.lang.Long	**getLastOccurrencesTime**(java.lang.String category, java.lang.String key) Retrieves last occurrence.
java.util.Vector	**getOccurences**(java.lang.String category, java.lang.String key) Retrieves all occurrences.
void	**init**(java.lang.String identifier, net.osbakk.pi.util.properties.Properties properties) Initialises the datastorage.
boolean	**isConnected**() Returns the current connection state.
void	**setData**(java.lang.String category, java.lang.String key, java.lang.Object data, java.lang.Long time) Stores data.

Method Detail

init

public void **init**(java.lang.String identifier,
net.osbakk.pi.util.properties.Properties properties)
throws java.lang.IllegalStateException,
java.lang.NullPointerException,

java.lang.Exception
Initialises the datastorage.
Parameters:
properties - the properties.
Throws:
java.lang.IllegalStateException - if already initialised.
java.lang.Exception - if an unexpected exception occurred.

---

connect
public void **connect**()
throws java.lang.IllegalStateException,
java.lang.Exception
Activates the datastorage.
Throws:
java.lang.IllegalStateException - if already connected or not initialised.
java.lang.Exception - if an unexpected exception occurred.

---

disconnect
public void **disconnect**()
throws java.lang.IllegalStateException,
java.lang.Exception
Inactivates the datastorage
Throws:
java.lang.IllegalStateException - if not connected.
java.lang.Exception - if an unexpected exception occurred.

---

isConnected
public boolean **isConnected**()
Returns the current connection state.
Returns:
connection state (true/false).

---

setData
public void **setData**(java.lang.String category,
java.lang.String key,
java.lang.Object data,
java.lang.Long time)
throws java.lang.IllegalStateException,
java.lang.NullPointerException,
java.lang.Exception
Stores data. Adds the provided piece of information to the datastore. Note:
- The category + key always yield a unique key.
- The data is always a serializable object.
- The timestamp is represented as seconds relative to 1 January 1970 UTC.
- If the timestamp is null the current time must be used.
- If the history length has been reached, or historic data is not supported, the oldest data item must be removed to make space.
Parameters:
category - The data category.

key - The key.
data - The data.
time - The timestamp or null.
Throws:
java.lang.IllegalStateException - if not connected.
java.lang.NullPointerException - if category, key, or data is null.
java.lang.Exception - if an unexpected exception occurred.

---

getData
public java.lang.Object **getData**(java.lang.String category,
java.lang.String key,
java.lang.Long time)
throws java.lang.IllegalStateException,
java.lang.NullPointerException,
java.lang.Exception
Retrieves data. Retrieves the requested piece of information from the datastore. Note:
- The category + key always yield a unique key
- The data is always a serializable object
- The timestamp is represented as seconds relative to 1 January 1970 UTC.
- If the timestamp is null the most recent occurrence must be retrieved.
- If the data item does not exist then null must be returned.
Parameters:
category - The data category.
key - The key.
time - The timestamp or null.
Returns:
The data item or null.
Throws:
java.lang.IllegalStateException - if not connected.
java.lang.NullPointerException - if category or key is null.
java.lang.Exception - if an unexpected exception occurred.

---

deleteData
public void **deleteData**(java.lang.String category,
java.lang.String key,
java.lang.Long time)
throws java.lang.IllegalStateException,
java.lang.NullPointerException,
java.lang.Exception
Deletes data. Deletes a previously stored data item. Note:
- The category + key always yield a unique key
- The timestamp is represented as seconds relative to 1 January 1970 UTC.
- If the timestamp is null the most recent occurrence must be deleted.
Parameters:
category - The data category.
key - The key.
time - The timestamp or null.
Throws:

java.lang.IllegalStateException - if not connected.
java.lang.NullPointerException - if category or key is null.
java.lang.Exception - if an unexpected exception occurred.

---

deleteData
public void **deleteData**(java.lang.String category,
java.lang.String key)
throws java.lang.IllegalStateException,
java.lang.NullPointerException,
java.lang.Exception
Deletes data. Deletes all occurrences of a certain category + key. Note:
- The category + key always yield a unique key
Parameters:
category - The data category.
key - The key.
Throws:
java.lang.IllegalStateException - if not connected.
java.lang.NullPointerException - if category or key is null.
java.lang.Exception - if an unexpected exception occurred.

---

deleteData
public void **deleteData**(java.lang.String category)
throws java.lang.IllegalStateException,
java.lang.NullPointerException,
java.lang.Exception
Deletes data. Deletes a stored data category with all its keys and occurrences.
Parameters:
category - The data category.
Throws:
java.lang.IllegalStateException - if not connected.
java.lang.NullPointerException - if category is null.
java.lang.Exception - if an unexpected exception occured.

---

deleteData
public void **deleteData**()
throws java.lang.IllegalStateException,
java.lang.Exception
Deletes data. Deletes all stored data.
Throws:
java.lang.IllegalStateException - if not connected.
java.lang.Exception - if an unexpected exception occured.

---

getKeysTable
public java.util.Hashtable **getKeysTable**(java.lang.String category)
throws java.lang.IllegalStateException,
java.lang.NullPointerException,
java.lang.Exception
Retrieves keys. Retrieves the keys and timestamps for a given category. Note:

- The keys must be returned in a Hashtable with keys as keys and the latest update time as values.
- The update timestamp must be represented as seconds relative to 1 January 1970 UTC.
- If the category does not exist null must be returned.

Parameters:
category - The data category.

Returns:
The keys and last update times or null.

Throws:
java.lang.IllegalStateException - if not connected.
java.lang.NullPointerException - if category is null.
java.lang.Exception - if an unexpected exception occurred.

---

getOccurences

public java.util.Vector **getOccurences**(java.lang.String category,
java.lang.String key)
throws java.lang.IllegalStateException,
java.lang.NullPointerException,
java.lang.Exception

Retrieve occurrences. Retrieve the occurrences for a given category + key.
Note:
- The category + key always yield a unique key
- The occurrences must be returned as timestamps in a Vector.
- The timestamp must be represented as seconds relative to 1 January 1970UTC.
- If the category + key do not exist null must be returned.

Parameters:
category - The data category.
key - The key.

Returns:
The occurences or null.

Throws:
java.lang.IllegalStateException - if not connected.
java.lang.NullPointerException - if category or key is null.
java.lang.Exception - if an unexpected exception occurred.

---

getKeys

public java.util.Vector **getKeys**(java.lang.String category)
throws java.lang.IllegalStateException,
java.lang.NullPointerException,
java.lang.Exception

Retrieves the keys for a given category. Note:
- The keys must be returned in a Vector.
- If the category does not exist null must be returned.

Parameters:
category - The data category.

Returns:
The keys or null.

Throws:

java.lang.IllegalStateException - if not connected.
java.lang.NullPointerException - if category is null.
java.lang.Exception - if an unexpected exception occured.

---

getCategories
public java.util.Vector **getCategories**()
throws java.lang.IllegalStateException,
java.lang.Exception
Retrieves all categories. Note:
- The categories must be returned in a Vector.
- If no category exist null must be returned.
Returns:
The categories or null.
Throws:
java.lang.IllegalStateException - if not connected.
java.lang.NullPointerException - if category is null.
java.lang.Exception - if an unexpected exception occured.

---

flush
public void **flush**()
throws java.lang.IllegalStateException,
java.lang.Exception
Flushes queue. Flushes the data in the queue. After being called changes are guaranteed to be permanent. Note:
- The data in the queue must be saved/deleted permanently when called.
Throws:
java.lang.IllegalStateException - if not connected.
java.lang.Exception - if an unexpected exception occurred.

---

clear
public void **clear**()
throws java.lang.IllegalStateException,
java.lang.Exception
Clear datastorage. Clears the datastorage.
Throws:
java.lang.IllegalStateException - if not connected.
java.lang.Exception - if an unexpected exception occurred.

---

getLastOccurrencesTime
public java.lang.Long **getLastOccurrencesTime**(java.lang.String category,
java.lang.String key)
throws java.lang.IllegalStateException,
java.lang.NullPointerException,
java.lang.Exception
Retrieves the last occurrence. Gets the time of the last occurrence of a category + key combination. Note:
- The category + key always yield a unique key
- The timestamp must be specified as seconds relative to 1 January 1970 UTC.
- If the category + key do not exist null must be returned.

Parameters:
category - The data category. * @param key The key.
Returns:
The occurrence timestamp or null.
Throws:
java.lang.IllegalStateException - if not connected.
java.lang.NullPointerException - if category or key is null.

C.2 TCP interface

---

net.osbakk.pi.cm.client
Interface TCPInterface

---

public interface **TCPInterface**
The TCPInterface defines the methods the Transport Client Plug-in classes must implement.

Method Summary	
void	**init**(java.lang.String address, net.osbakk.pi.util.properties.Properties properties) Initialises the Transport Client Plug-in.
java.lang.String	**post**(java.lang.String message) Posts a 'message' to the recipient.

Method Detail

init
public void **init**(java.lang.String address,
net.osbakk.pi.util.properties.Properties properties)
throws java.lang.IllegalStateException,
java.lang.NullPointerException,
java.lang.Exception
Initialises the Transport Client Plug-in. Note:
- One instance of the TCP is created per recipient.
- The format of the recipient's address is plug-in specific.
Parameters:
address - The address.
properties - the properties.
Throws:
java.lang.IllegalStateException - if already initialised.
java.lang.NullPointerException - if address or properties is null.
java.lang.Exception - if an unexpected exception occurred.

---

post
public java.lang.String **post**(java.lang.String message)
throws java.lang.IllegalStateException,
java.lang.NullPointerException,

java.lang.Exception
Posts a 'message' to the recipient. Note:
- The message is encoded using the communication format.
- The response must be encoded using the communication format.
- The communication format is based on CC/PP. See separate description for full details.

Parameters:
message - The message.

Returns:
The response.

Throws:
java.lang.IllegalStateException - if not initialised.
java.lang.NullPointerException - if message is null.
java.lang.Exception - if an unexpected exception occurred.

C.3 TSP interface

---

net.osbakk.pi.cm.server
Interface TSPInterface

---

public interface **TSPInterface**
The TSPInterface defines the methods the Transport Server Plug-in classes must implement.

Method Summary	
void	**addComEventListener**(net.osbakk.pi.cm.server.ComEventListener listener) Adds a communication event listener.
void	**connect**() Activates the Transport Server Plug-in.
void	**disconnect**() Inactivates the Transport Server Plug-in.
java.lang.String[]	**getAddresses**() Retrieves the addresses associated with this TSP plug-in.
void	**init**(net.osbakk.pi.util.properties.Properties properties) Initialises the Transport Server Plug-in.
boolean	**isConnected**() Returns the current connection state.
void	**reply**(ComEvent event) Responds to a request.

Method Detail

init

public void **init**(net.osbakk.pi.util.properties.Properties properties)
throws java.lang.IllegalStateException,
java.lang.NullPointerException,
java.lang.Exception
Initialises the Transport Server Plug-in. Note:
- One instance of the TSP plug-in is created per protocol.
Parameters:
properties - the properties.
Throws:
java.lang.IllegalStateException - if already initialised.
java.lang.NullPointerException - if properties is null.
java.lang.Exception - if an unexpected exception occurred.

---

connect
public void **connect**()
throws java.lang.IllegalStateException,
java.lang.Exception
Activates the Transport Server Plug-in.
Throws:
java.lang.IllegalStateException - if already connected or not initialised.
java.lang.Exception - if an unexpected exception occurred.

---

disconnect
public void **disconnect**()
throws java.lang.IllegalStateException,
java.lang.Exception
Inactivates the Transport Server Plug-in.
Throws:
java.lang.IllegalStateException - if not connected.
java.lang.Exception - if an unexpected exception occurred.

---

isConnected
public boolean **isConnected**()
Returns the current connection state.
Returns:
connection state (true/false).

---

getAddresses
public java.lang.String[] **getAddresses**()
throws java.lang.IllegalStateException,
java.lang.Exception
Retrieves the addresses associated with this TSP plug-in. Note:
- The addresses must be returned as an array of Strings, where each address is one entry.
- The format of the addresses is plug-in specific.
Returns:
the addresses.
Throws:
java.lang.IllegalStateException - if not connected.

java.lang.Exception - if an unexpected exception occurred.

---

reply
public void **reply**(ComEvent event)
throws java.lang.IllegalStateException,
java.lang.NullPointerException,
java.lang.Exception
Responds to a request. Note:
- The response to the request is contained with in the event.
Parameters:
event - The event.
Throws:
java.lang.IllegalStateException - if not connected.
java.lang.NullPointerException - if event is null.
java.lang.Exception - if an unexpected exception occurred.

---

addComEventListener
public void
addComEventListener(net.osbakk.pi.cm.server.ComEventListener listener)
throws java.lang.IllegalStateException,
java.lang.NullPointerException,
java.lang.Exception
Adds a communication event listener. Note:
- Multiple concurrent listeners must be supported.
- Upon receiving a request the TSP plug-in must create an event and call the listener's 'comEventOccured' method with it.
Parameters:
listener - The listener.
Throws:
java.lang.IllegalStateException - if not connected.
java.lang.NullPointerException - if listener is null.
java.lang.Exception - if an unexpected exception occurred.

C.4 Communication event listener

---

net.osbakk.pi.cm.server
Interface ComEventListenerInterface
All Superinterfaces:
java.util.EventListener

---

public interface **ComEventListenerInterface**
extends java.util.EventListener
The ComEventListenerInterface defines the methods the ComEventListener implement.

Method Summary

void	**comEventOccured**(ComEvent event) Dispatches an event to the Context Manager.

Method Detail

comEventOccured
public void **comEventOccured**(ComEvent event)
throws java.lang.NullPointerException,
java.lang.Exception
Dispatches an event to the Context Manager.
Parameters:
event - The event.
Throws:
java.lang.NullPointerException - if event is null.
java.lang.Exception - if an unexpected exception occurred.

C.5 Communication event

---

net.osbakk.pi.cm.server
Class ComEvent
java.lang.Object
|
+--**net.osbakk.pi.cm.server.ComEvent**

---

public class **ComEvent**
extends java.lang.Object
The ComEvent class is a wrapper for events, i.e requests and responses.

Constructor Summary
ComEvent(java.lang.String message) Constructs a Communication Event.

Method Summary	
java.lang.Object	**getEventDetails**() Sets the event details.
java.lang.String	**getMessage**() Retrieves the message.
void	**setEventDetails**(java.lang.Object details) Sets the event details.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ComEvent

public **ComEvent**(java.lang.String message)
throws java.lang.NullPointerException

Constructs a Communication Event. Note:
- The message holds the request/response. It is encoded using the communication format.
- The communication format is based on CC/PP. See separate description for full details.

Parameters:

message - The message.

Throws:

java.lang.NullPointerException - if message is null.

Method Detail

getMessage

public java.lang.String **getMessage**()

Retrieves the message. Note:
- The message holds the request/response. It is encoded using the communication format.
- The communication format is based on CC/PP. See separate description for full details.

Returns:

the message.

---

setEventDetails

public void **setEventDetails**(java.lang.Object details)

Sets the event details Note:
- The details of the event are protocol specific.
- The details may be null.

Parameters:

details - The details.

---

getEventDetails

public java.lang.Object **getEventDetails**()

Sets the event details Note:
- The details of the event are protocol specific.
- The details may be null.

Returns:

The details.

C.6 EDP Interface

net.osbakk.pi.cm.cryptography
Interface EDPInterface

public interface **EDPInterface**
The EDPInterface defines the methods the EDP classes must implement.

Method Summary	
void	**decrypt**(java.io.InputStream input, java.io.OutputStream output, byte[] privateKey) Decrypts data using the provided key.
void	**encrypt**(java.io.InputStream input, java.io.OutputStream output, byte[] publicKey) Encrypts data using the provided key.
Keypair	**generateKeys**() Generates a new set of keys.
void	**init**(net.osbakk.pi.util.properties.Properties properties) Initialises the Encryption Decryption Plug-in.

Method Detail

init
public void **init**(net.osbakk.pi.util.properties.Properties properties)
throws java.lang.NullPointerException,
java.lang.IllegalStateException,
java.lang.Exception
Initialises the Encryption Decryption Plug-in. Note:
- One instance of the EDP plug-in is created per Plug-in.
Parameters:
properties - the properties.
Throws:
java.lang.IllegalStateException - if already initilised.
java.lang.NullPointerException - if properties is null.
java.lang.Exception - if an unexpected exception occurred.

---

generateKeys
public Keypair **generateKeys**()
throws java.lang.IllegalStateException,
java.lang.Exception
Generates a new set of keys. Note:
- The generated keys must be returned as a keypair.
Returns:
the keypair.
Throws:
java.lang.IllegalStateException - if not initialised.
java.lang.Exception - if an unexpected exception occurred.

---

encrypt
public void **encrypt**(java.io.InputStream input,
java.io.OutputStream output,
byte[] publicKey)
throws java.lang.NullPointerException,
java.lang.IllegalStateException,
java.lang.Exception
Encrypts data using the provided key. Note:
- The unencrypted data, to encrypt, is provided with an inputstream.
- The encrypted data must be returned with an outputstream.
- The public key refers to the key required for encryption.
Parameters:
input - the inputstream.
output - the outputstream.
publicKey - the public key.
Throws:
java.lang.IllegalStateException - if not initialised.
java.lang.NullPointerException - if input, output, or publicKey is null.
java.lang.Exception - if an unexpected exception occurred.

---

decrypt
public void **decrypt**(java.io.InputStream input,
java.io.OutputStream output,
byte[] privateKey)
throws java.lang.NullPointerException,
java.lang.IllegalStateException,
java.lang.Exception
Decrypts data using the provided key. Note:
- The encrypted data, to decrypt, is provided with an inputstream.
- The decrypted data must be returned with an outputstream.
- The private key refers to the key required for decryption.
Parameters:
input - the inputstream.
output - the outputstream.
privateKey - the private key.
Throws:
java.lang.IllegalStateException - if not initialised.
java.lang.NullPointerException - if input, output, or privateKey is null.
java.lang.Exception - if an unexpected exception occurred.

C.7 Key pair

---

net.osbakk.pi.cm.cryptography
Class Keypair
java.lang.Object
|
+--**net.osbakk.pi.cm.cryptography.Keypair**
All Implemented Interfaces:

java.io.Serializable

---

public class **Keypair**
extends java.lang.Object
implements java.io.Serializable
The Keypair class is a wrapper for private and public keys.
See Also:
Serialized Form

Constructor Summary
Keypair(byte[] privateKey, byte[] publicKey) Constructs a Keypair.

Method Summary	
byte[]	**getPrivateKey**() Retrieves the private key.
byte[]	**getPublicKey**() Retrieves the public key.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Keypair
public **Keypair**(byte[] privateKey,
byte[] publicKey)
throws java.lang.NullPointerException
Constructs a Keypair. Note:
- The private key refers to the key used for decryption.
- The public key refers to the key used for encryption.
Parameters:
privateKey - the private key.
publicKey - the public key.
Throws:
java.lang.NullPointerException - if privateKey or publicKey is null.

Method Detail

getPrivateKey
public byte[] **getPrivateKey**()
Retrieves the private key. Note:
- The private key refers to the key used for decryption.
Returns:
the private key.

---

getPublicKey
public byte[] **getPublicKey**()
Retrieves the public key. Note:
- The public key refers to the key used for encryption.
Returns:
the public key.

C.8 REP Interface

---

net.osbakk.pi.cm.server
Interface REPInterface

---

public interface **REPInterface**
The REPInterface defines the methods the Resource Extension Plug-ins classes must implement.

Method Summary	
void	**init**(java.util.Hashtable arguments, CMLink cmServiceLink, net.osbakk.pi.util.properties.Properties properties) Initialises the Resource Extension Plug-in.
void	**perform**(java.lang.String contextItem) Invokes the resource extension for this request.
void	**start**() Starts the resource extension in the background.

Method Detail

init
public void **init**(java.util.Hashtable arguments,
CMLink cmServiceLink,
net.osbakk.pi.util.properties.Properties properties)
throws java.lang.IllegalStateException,
java.lang.Exception
Initialises the Resource Extension Plug-in.
Parameters:
arguments - The arguments.
cmServiceLink - The cmservicelink.
properties - The properties.
Throws:
java.lang.IllegalStateException - if already initialised.
java.lang.Exception - if an unexpected exception occured.

---

perform
public void **perform**(java.lang.String contextItem)
throws java.lang.IllegalStateException,
java.lang.Exception

Invokes the resource extension for this request.
Parameters:
contextItem - The context item prompting the action, or null if not disclosed.
Throws:
java.lang.IllegalStateException - if not initialised.
java.lang.Exception - if an unexpected exception occurred.

---

start
public void **start**()
throws java.lang.IllegalStateException,
java.lang.Exception
Starts the resource extension in the background.
Throws:
java.lang.IllegalStateException - if not initialised.
java.lang.Exception - if an unexpected exception occurred.

C.9 Context manager link

---

net.osbakk.pi.cm.server
Class CMLink
java.lang.Object
|
+--**net.osbakk.pi.cm.server.CMLink**

---

public final class **CMLink**
extends java.lang.Object
The CMLink class provides an internal communication link to the context manager.

Constructor Summary
CMLink(net.osbakk.pi.cm.server.ContextManager contextManager) Constructs a context manager link.

Method Summary	
net.osbakk.pi.cm.common.Message	**postRequest**(java.lang.String message) Posts a message.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

CMLink
public **CMLink**(net.osbakk.pi.cm.server.ContextManager contextManager)

throws java.lang.NullPointerException
Constructs a context manager link.
Parameters:
contextManager - The context manager.
Throws:
java.lang.NullPointerException - if contextManager is null.

Method Detail

postRequest
public net.osbakk.pi.cm.common.Message
postRequest(java.lang.String message)
throws java.lang.NullPointerException
Posts a message. The message is sent to the context manager for processing.
Parameters:
message - The message.
Returns:
the context manager's reply.
Throws:
java.lang.NullPointerException - if message is null.
java.lang.Exception - if an unexpected exception occurred.

D Configuring access

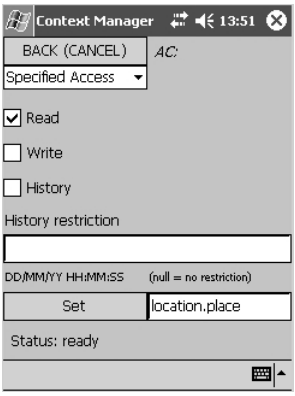
D.1 Scenario

Assume that Alice wants to allow her friend, Bob, to retrieve basic location information from her context manager. To enable this she needs to set up an account for Bob with read access to location.place. Since it is likely she will want to allow other friends to retrieve the same information the access should be defined using a role.

D.2 Procedure

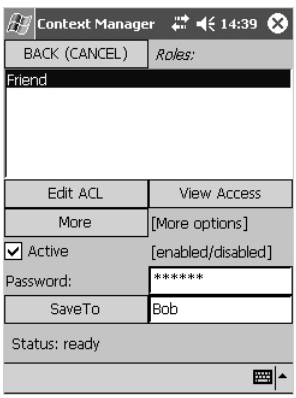
The administration console is used to configure access to the context manager. The procedure can be broken down into three steps.

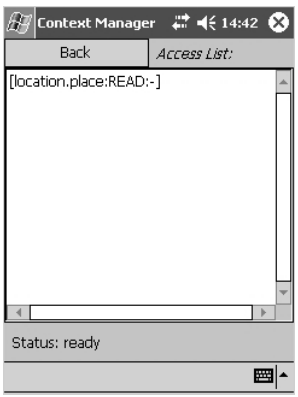
Firstly, an access control list (ACL) for basic location information must be created. For the scenario outlined, the ACL only needs to contain a single access control (AC), location.place, that grants read access to the corresponding context element. The screenshots below show, from left to right, the ACL for basic location being created and the AC for location.place being defined.



Secondly, a role for friends must be created. According to the scenario the role should grant read access to basic location information. Thus, the role needs to be associated with the access control list for basic location. The screenshot below shows the role Friend being defined.

Finally, a user account for Bob must be created. The account should be associated with the role Friend and specify an initial password for authentication. To enable the account to be used it must also be activated. The screenshots below show, from left to right, Bob's account being created and the resulting access being granted.





E Privacy survey

E.1 Questions

Introduction

This study is done as part of my research as a PhD student at the University of Kent where I research privacy protection for context-aware systems. Context-aware systems use information about us and our environments to provide enhanced and new services. One example of an existing commercial application is a location service that allow us to find nearby places and people. Location, though, is only one of many pieces of information that may be used by a context-aware system, others include: activity, contact details, weather information etc. The research I undertake aims to improve the level of privacy users of such system enjoy by allowing privacy preferences to be expressed and enforced. For example you may want to allow some people to know where you are but not others.

The purpose of this study is to evaluate two different protection mechanisms that allow privacy preferences to be described. It should only take approximately 10 minutes to complete. All the information is collected anonymously and will not be personally identifiable. The results may be published.

If you have any questions do not hesitate to contact me.

Thank you,

Patrik Osbakk
Computing Laboratory
University of Kent
Canterbury, Kent
CT2 7NF

pjo2@kent.ac.uk

Background

Age: ___

Gender: ○Male ○Female

How comfortable are you using computers?

○Very comfortable ○Comfortable ○Neither comfortable nor uncomfortable
○Uncomfortable ○Very uncomfortable

How comfortable are you using the Internet?

○Very comfortable ○Comfortable ○Neither comfortable nor uncomfortable
○Uncomfortable ○Very uncomfortable

How concerned are you about your privacy in real-life (your right to control the flow of your personal information)?

○Very concerned ○Concerned ○Neither concerned nor unconcerned
○Unconcerned ○Very unconcerned

How concerned are you about your privacy when online (your right to control the flow of your personal information)?

○Very concerned ○Concerned ○Neither concerned nor unconcerned
○Unconcerned ○Very unconcerned

Scenario

Assume that your mobile phone connects you to a context-aware system that collects and monitors your context, e.g. your location and current activity. This information is used to provide a number of enhanced services to you. For example when you are busy, e.g. in a meeting, calls can automatically be forwarded to your voicemail or the callers can be asked to call back later when convenient for you. Other services would allow you to find nearby places such as restaurants, shops, sights, etc. or even to be interactively guided to a named place. Furthermore imagine that your context information is shared with others, allowing you to both be located and to be able to locate others.

Given this scenario:

Who of the following should be allowed to access your location (Tick all the appropriate)?

□ Family member, □ Friend, □ Colleague
□ Boss/Supervisor/Teacher, □ Anyone (public)

Please tick the box(es) of those you feel should be able to find out where you are when they desire.

Who of the following should be allowed to access your activity (Tick all the appropriate)?

□ Family member, □ Friend, □ Colleague
□ Boss/Supervisor/Teacher, □ Anyone (public)

Please tick the box(es) of those you feel should be able to find out where you are when they desire.

Who of the following should be allowed to access your contact details (Tick all the appropriate)?

□ Family member, □ Friend, □ Colleague
□ Boss/Supervisor/Teacher, □ Anyone (public)

Please tick the box(es) of those you feel should be able to find out where you are when they desire.

Mechanisms

Continue to assume that you are connected to the context-aware system previously described and that it shares your context information with others. Two different protection mechanisms will now be presented and you are asked to set them up to match your previously indicated preferences as close as possible.

Protection Mechanism 1: Classification and Clearance Scheme - CCS

This mechanism focuses on the sensitivity of information and works like this: You tell the system how sensitive (private) you feel a piece of information is,

i.e. you classify it. You also tell the system how sensitive (private) information different people should be able to retrieve, i.e. you give people a clearance level. The system can then use this information to ensure that only those you want can find out where you are, what you are doing, and how to contact you.

Here each piece of context is classified on a scale of 0 to 100, where 0 is the most insensitive (public) and 100 is the most sensitive (private). Information that anyone should be able to find out should be classified with 0. Similarly each person that should be able to retrieve information is given a clearance level on a scale of 0 (only allowed to access public information) to 100 (can access everything). Please note that information is shared if the person has a clearance equal to or greater than the corresponding classification.

Please classify the context pieces below:

Location: ___
Activity: ___
Contact Details: ___

Please enter a value between 0 (public) and 100 (private) that represent how sensitive you feel the information is.

Please assign the people below the desired clearance (Anyone always has a level of 0, thus it cannot be changed):

Family member: ___
Friend: ___
Colleague: ___
Boss/supervisor/teacher: ___
Anyone (public): ___

Please enter a value between 0 (public) and 100 (private) that represent how sensitive information you feel that person should be able to retrieve.

How similar is this mechanism to how you reason about privacy day to day?

○Very similar ○Similar ○Neither similar nor unsimilar
○Unsimilar ○Very unsimilar}

How accurately do you feel you have been able to express your preferences using this mechanism?

○Very accurately ○Accurately ○Neither accurately nor inaccurately
○Inaccurately ○Very inaccurately

How easy did you find expressing your preferences using this mechanism?

○Very easy ○Easy ○Neither easy nor difficult ○Difficult ○Very difficult

Protection Mechanism 2: Role Based Access Control - RBAC

This mechanism focuses on what role a person has, where every role allows certain pieces of context information to be shared. The mechanism works like this: You first identify what roles you need, i.e. what roles do people around you have. You then tell the system what information a person having a specific role should be able to retrieve. After this you tell the system what roles people have. The system can then use this information to ensure that only those you want can find out where you are, what you are doing, and how to contact you.

Here each role can allow a combination of location, activity, and contact details to be retrieved including all and none of them. The roles are labelled Role 1 through to Role 5 so you are free to think of a role, e.g. Role 1, to represent any role you would like, e.g. friends. In this survey you can setup at most 5 roles. If you don't need all of them just leave the ones you don't need blank. Each person that should be able to retrieve information then has one or more roles associated. Please note that information is shared using the roles in the best possible way so that a person that has multiple roles can access the 'sum' of the roles.

Please setup what information should be shared with each role:

Role 1: □ Location, □ Activity, □ Contact Details
Role 2: □ Location, □ Activity, □ Contact Details
Role 3: □ Location, □ Activity, □ Contact Details
Role 4: □ Location, □ Activity, □ Contact Details
Role 5: □ Location, □ Activity, □ Contact Details

Please tick the box(es) of a particular role to allow the information in question to be released. For example a tick in the location box of Role1 would later allow anyone associated with Role1 to retrieve where you are.

Please assign roles to the following persons:

A family member:	□ Role1, □ Role2, □ Role3, □ Role4, □ Role5
A friend:	□ Role1, □ Role2, □ Role3, □ Role4, □ Role5
A colleague:	□ Role1, □ Role2, □ Role3, □ Role4, □ Role5
Your boss/supervisor/teacher:	□ Role1, □ Role2, □ Role3, □ Role4, □ Role5
Anyone (public):	□ Role1, □ Role2, □ Role3, □ Role4, □ Role5

Please tick the box(es) of a particular person to associate this person with the role in question. For example a tick in the Role1 box of A friend would allow this friend to retrieve the information Role1 allows.

How similar is this mechanism to how you reason about privacy day to day?

○Very similar ○Similar ○Neither similar nor unsimilar
○Unsimilar ○Very unsimilar

How accurately do you feel you have been able to express your preferences using this mechanism?

○Very accurately ○Accurately ○Neither accurately nor inaccurately
○Inaccurately ○Very inaccurately

How easy did you find expressing your preferences using this mechanism?

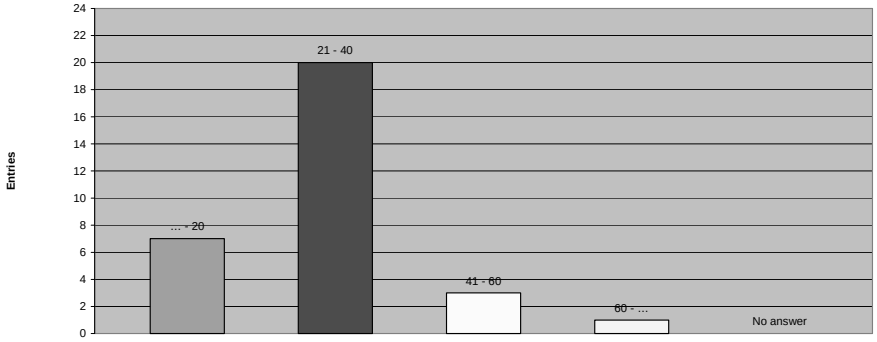
○Very easy ○Easy ○Neither easy nor difficult ○Difficult ○Very difficult

Submission

Please press the button below to submit the study.

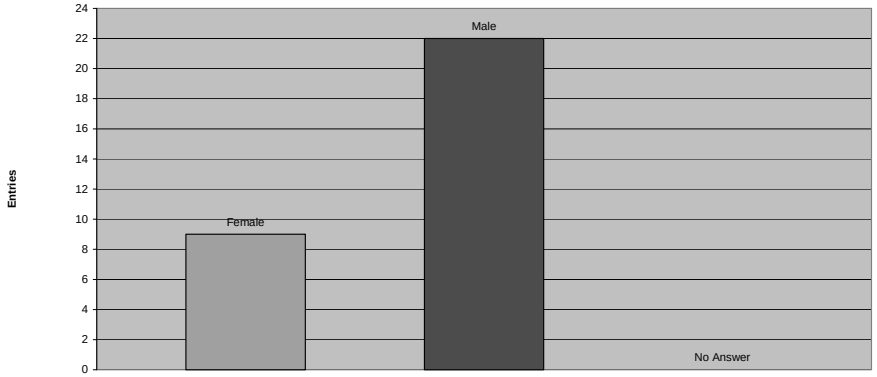
E.2 Responses

Age



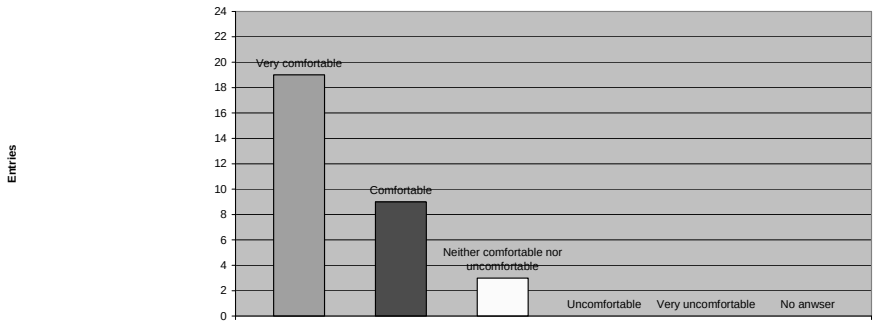
… - 20	7
21 - 40	20
41 - 60	3
60 - …	1
No answer	0

Gender



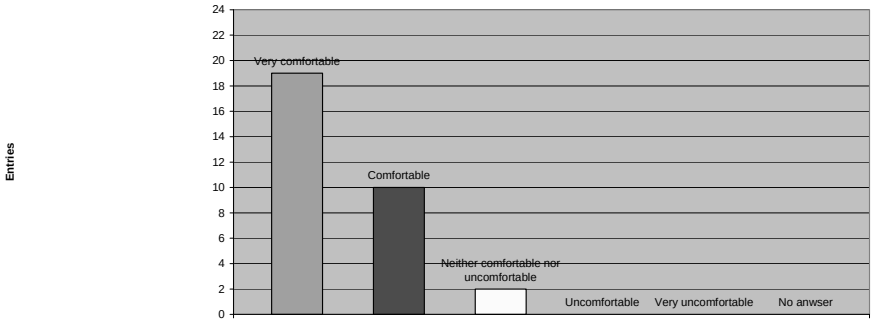
Female	9
Male	22
No Answer	0

How comfortable are you using computers?



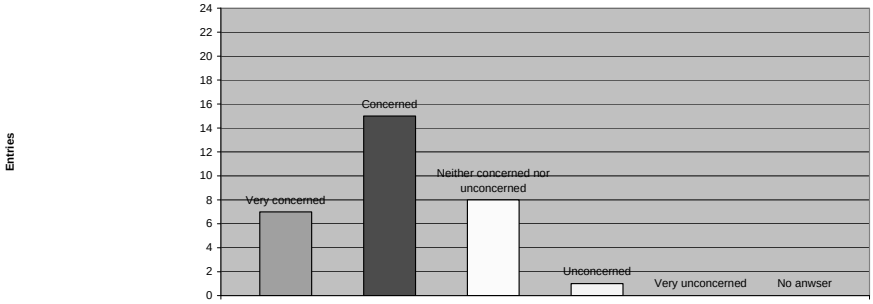
Very comfortable	19
Comfortable	9
Neither comfortable nor uncomfortable	3
Uncomfortable	0
Very uncomfortable	0
No anwser	0

How comfortable are you using the Internet?



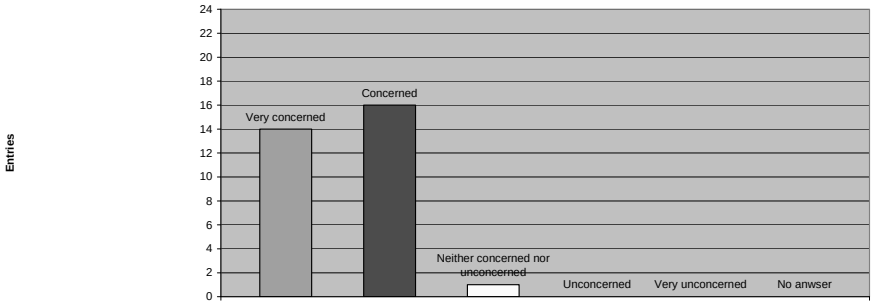
Very comfortable	19
Comfortable	10
Neither comfortable nor uncomfortable	2
Uncomfortable	0
Very uncomfortable	0
No anwser	0

How concerned are you about your privacy in real-life (your right to control the flow of your personal information)?



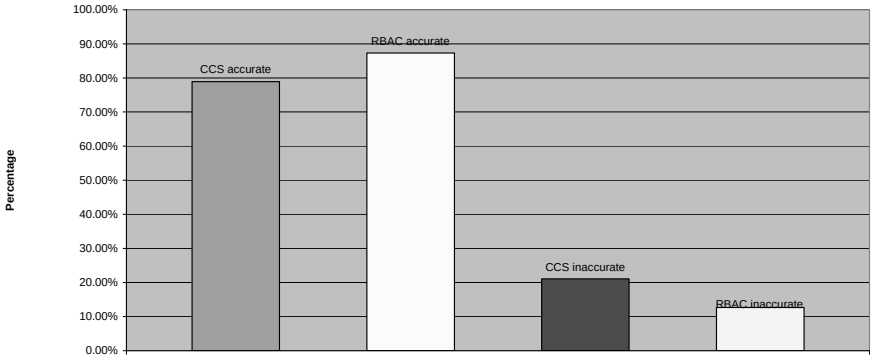
Very concerned	7
Concerned	15
Neither concerned nor unconcerned	8
Unconcerned	1
Very unconcerned	0
No anwser	0

How concerned are you about your privacy when online (your right to control the flow of your personal information)?



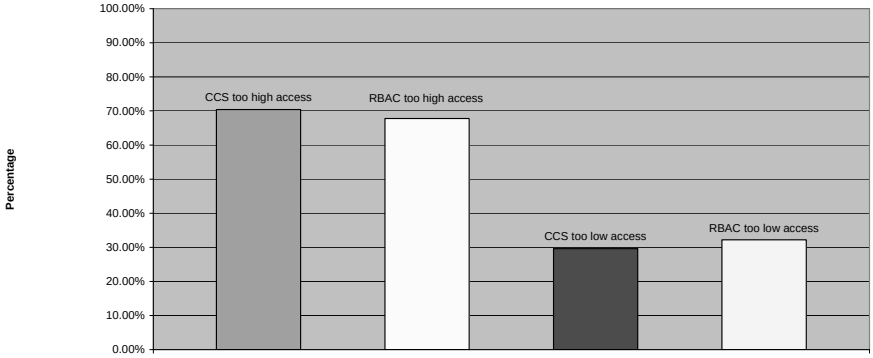
Very concerned	14
Concerned	16
Neither concerned nor unconcerned	1
Unconcerned	0
Very unconcerned	0
No anwser	0

Overall accuracy



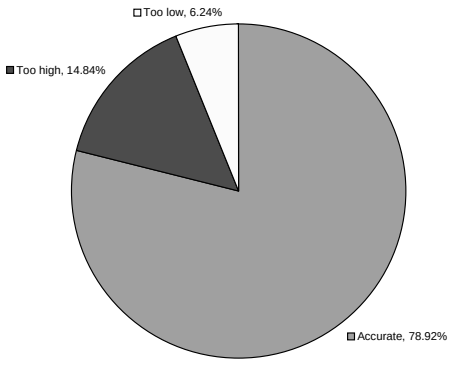
CCS accurate	78.92%
RBAC accurate	87.31%
CCS inaccurate	21.08%
RBAC inaccurate	12.69%

Overall effect of inaccuracy

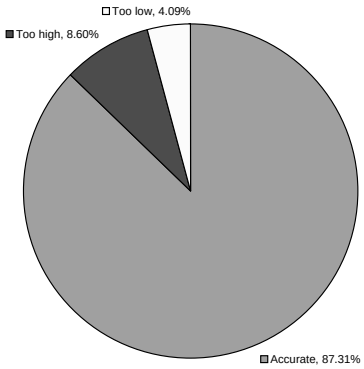


CCS too high access	70.41%
RBAC too high access	67.80%
CCS too low access	29.59%
RBAC too low access	32.20%

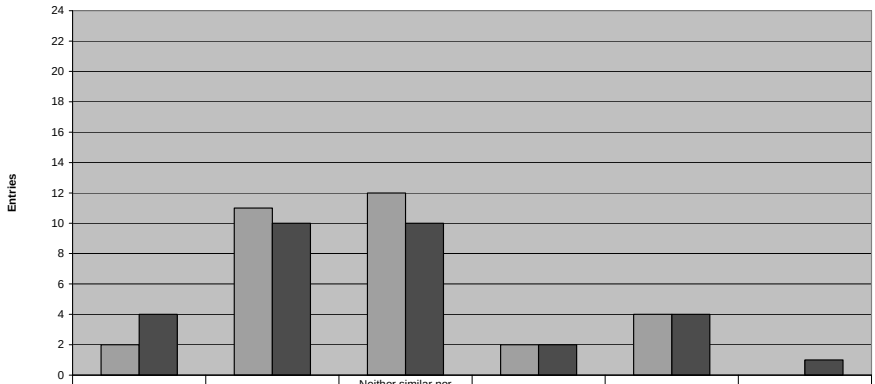
CCS Access



RBAC Access

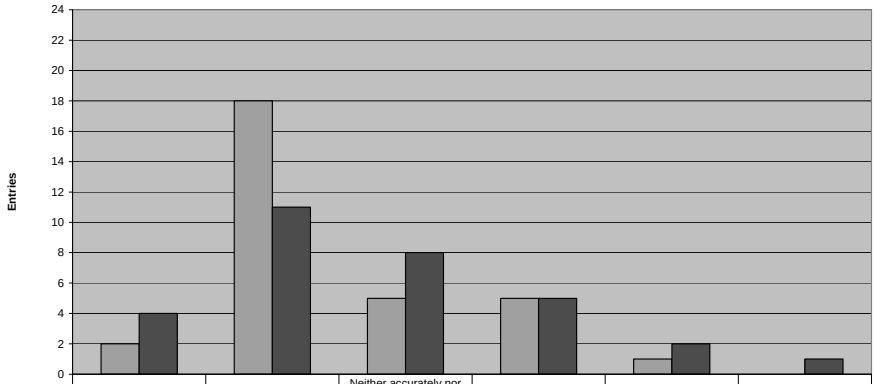


How similar is this mechanism to how you reason about privacy day to day?



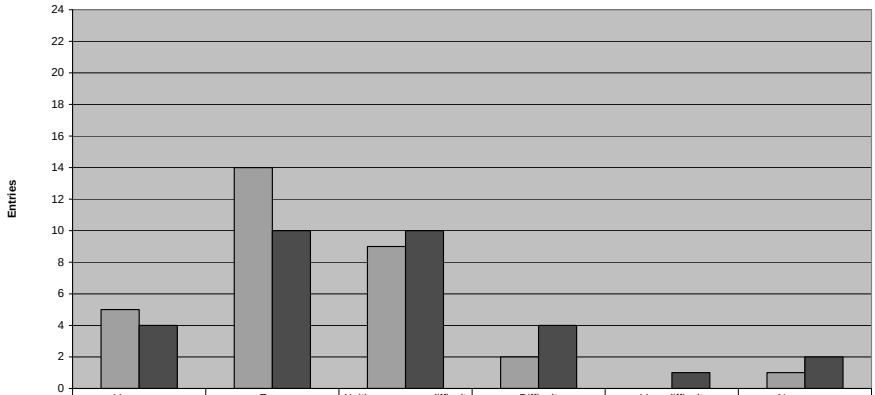
	Very similar	Similar	Neither similar nor unsimilar	Unsimilar	Very unsimilar	No anwser
CCS	2	11	12	2	4	0
RBAC	4	10	10	2	4	1

How accurately do you feel you have been able to express your preferences using this mechanism?



	Very accurately	Accurately	Neither accurately nor inaccurately	Inaccurately	Very inaccurately	No anwser
CCS	2	18	5	5	1	0
RBAC	4	11	8	5	2	1

How easy did you find expressing your preferences using this mechanism?



	Very easy	Easy	Neither easy nor difficult	Difficult	Very difficult	No anwser
CCS:	5	14	9	2		1
RBAC	4	10	10	4	1	2

BIBLIOGRAPHY

[Aadland, Angel et al. 2002]
Aadland, B., M. Angel, et al. (2002) kXML Project. Version 1.21. *Webpage*.
Available at: http://kxml.objectweb.org/index.html
Last accessed: 20/12/2005

[Abrantix 2005]
Abrantix AG (2005). P3P Editor. *Webpage*.
Available at: http://www.p3peditor.com
Last accessed: 27/12/2005

[Adams 1997]
Adams, S. (1997). The Dilbert Principle, *Boxtree*.

[Addlesee, Curwen 2001]
Addlesee, M., R. Curwen, et al. (2001). Implementing a Sentient Computing System. *IEEE Computer* **34**(8): pp. 50-56.
Available at: http://www-lce.eng.cam.ac.uk/publications/files/tr.2001.8.pdf
Last accessed: 26/06/2004

[BBC News 2005]
BBC News website (2005). Card fraudsters 'targeting web'. BBC News. *Webpage*.
Available at: http://news.bbc.co.uk/1/hi/business/4243137.stm
Last accessed: 27/12/2005

[Borenstein, Freed 1992]
Borenstein, N. and N. Freed (1992). MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies. *Webpage*.
Available at: http://www.ietf.org/rfc/rfc1341.txt
Last accessed: 20/12/2005

[Bouncy Castle 2003]
Legion of the Bouncy Castle (2003). The Bouncy Castle Crypto package (release 1.21). *Webpage*.
Available at: ftp://ftp.bouncycastle.org/pub/release1.21/
Last accessed: 21/12/2005

[Brostoff, Sasse et al. 2005]
Brostoff, S., M. A. Sasse, et al. (2005). R-What? Development of a Role-Based Access Control (RBAC) Policy-Writing Tool for e-Scientists. *Software: Practice and Experience* **35**(9): pp. 835-856.
Available at: http://www.cs.kent.ac.uk/pubs/2005/2275/content.pdf
Last accessed: 14/01/2006

[Brotherton, Abowd et al. 1999]
Brotherton, J. A., G. D. Abowd, et al. (1999). Supporting Capture and Access Interfaces for Informal and Oppertunistic Meetings. *GVU Technical Report; GIT-GVU-99-06*.
Available at: http://hdl.handle.net/1853/3373
Last accessed: 07/04/2007

[Brown, Bovey et al. 1997]
Brown, P. J., J. D. Bovey, et al. (1997). Context-aware applications: from the laboratory to the marketplace. *IEEE Personal Communications* **4**(5): pp. 58-64.

[Brown, Jones 2004]
Brown, P. J. and G. J. F. Jones (2004) Context-awareness and privacy: an inevitable clash? (Version #1). *Webpage*.
Available at: http://www.dcs.ex.ac.uk/~pjbrown/papers/ieee_privacy.pdf
Last accessed: 27/02/2004

[Butz, Beschers et al. 1998]
Butz, A., C. Beshers, et al. (1998). Of Vampire Mirrors and Privacy Lamps: Privacy Management in Multi-User Augmented Environments. *Proceedings of the 11th annual ACM symposium on User interface software and technology*, San Francisco, California, United States, ACM Press.
Available at: http://www1.cs.columbia.edu/~butz/publications/papers/uist98.pdf
Last accessed: 07/01/2006

[Chadwick, Otenko et al. 2003]
Chadwick, D. W., A. Otenko, et al. (2003). Role-Based Access Control With X.509 Attribute Certificates. *IEEE Internet Computing*. 7:2 pp. 62-69.

[Chadwick, Otenko 2004]
Chadwick, D. W. and A. Otenko (2004). Implementing Role Based Access Controls using X.509 Privilege Management - the PERMIS Authorisation Infrastructure. *NATO Advanced Networking Workshop on Advanced Security Technologies in Networking*, Bled, Slovenia.
Available at: http://www.cs.kent.ac.uk/pubs/2004/2279/content.pdf
Last accessed: 14/01/2006

[Chen, Kotz 2002]
Chen, G. and D. Kotz (2002). Solar: A pervasive-computing infrastructure for context-aware mobile applications. *Dartmouth Computer Science Technical Report*. TR2002-421.
Available at: ftp://ftp.cs.dartmouth.edu/TR/TR2002-421.pdf
Last accessed: 08/12/2005

[Cheverst, Davies et al. 2000]
Cheverst, K., N. Davies, et al. (2000). Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences. *Conference on Human Factors and Computing Systems*, The Hague, The Netherlands.
Available at: http://www.guide.lancs.ac.uk/CHIpaper.pdf
Last accessed: 13/12/2005

[Code Infusion 2005]
Code Infusion, LLC (2005). P3PEdit. *Webpage*.
Available at: http://p3pedit.com/
Last accessed: 27/12/2005

[Covington, Long et al. 2001]
Covington, M. J., W. Long, et al. (2001). Securing Context-Aware Applications Using Environment Roles. *Proceedings of the Sixth ACM Symposium on Access control models and technologies*, Chantilly, Virginia, USA, ACM Press.

[Cranor, Byers et al. 2003]
Cranor, L., S. Byers, et al. (2003). An Analysis of P3P Deployment on Commercial, Government, and Children's Web Sites as of May 2003. *Technical Report prepared for the 14 May 2003 Federal Trade Commission Workshop on Technologies for Protecting Personal Information.*
Available at: http://www.research.att.com/projects/p3p/p3p-census-may03.pdf
Last accessed: 20/12/2005

[Crowcroft 2003]
Crowcroft, J. (2003). Scalable Ubiquitous Computing Systems or just Ubiquitous Systems. A 15-year Grand Challenge for Computer Science. *Webpage*.
Available at:
http://www.nesc.ac.uk/esi/events/Grand_Challenges/proposals/US.pdf
Last accessed: 08/12/2005

[Dallas Semiconductor, Maxim 2005A]
Dallas Semiconductor and Maxim (2005). DSTINIm400 Networked Microcontroller Evaluation Board. Rev: 091605. *Webpage*.
Available at: http://pdfserv.maxim-ic.com/en/ds/DSTINIM400.pdf
Last accessed: 14/12/2005

[Dallas Semiconductor, Maxim 2005B]
Dallas Semiconductor and Maxim (2005). What is an iButton?. *Webpage*.
Available at: http://www.maxim-ic.com/products/ibutton/ibuttons/index.cfm
Last accessed: 21/12/2005

[Debaty, Caswell 2000]
Debaty, P. and D. Caswell (2000). Uniform Web Presence Architecture for People, Places, and Things. *HP Labs Technical Report* HPL-2000-67.
Available at: http://www.hpl.hp.com/techreports/2000/HPL-2000-67.html
Last accessed: 13/04/2004

[Debaty, Goddi et al. 2003]
Debaty, P., P. Goddi, et al. (2003). Integrating the Physical World with the Web to Enable Context-Enhanced Services. *HP Labs Technical Report* HPL-2003-192.
Available at: http://www.hpl.hp.com/techreports/2003/HPL-2003-192.html
Last accessed: 13/04/2004

[Dey, Abowd 2000A]
Dey, A. K. and G. D. Abowd (2000). Towards a Better Understanding of Context and Context-Awareness. *Workshop on The What, Who, Where, When, and How of Context-Awareness, 2000 Conference on Human Factors in Computing Systems*, The Hague, The Netherlands.
Available at: ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/99-22.pdf
Last accessed: 20/06/2004

[Dey 2000B]
Dey, A. K. (2000). Providing Architectural Support for Building Context-Aware Applications. *Graphics, Visualization and Usability Center, College of Computing*. Atlanta, US, Georgia Institute of Technology**.**
Available at: http://www.cc.gatech.edu/fce/ctk/pubs/dey-thesis.pdf
Last accessed: 03/12/2005

[Dey, Abowd 2000C]
Dey, A. K. and G. D. Abowd (2000). The Context Toolkit: Aiding the development of Context-Aware Applications. *Workshop on Software Engineering for Wearable and Pervasive Computing*.
Available at: http://www.cc.gatech.edu/fce/contexttoolkit/pubs/SEWPC00.pdf
Last accessed: 03/12/2005

[Duan 2002]
Duan, M. (2002). An Introduction to Art, the Wireless Way. *mpulse magazine*. **2002**:October.
Available at: http://www.cooltown.com/cooltown/mpulse/1002-lasarsegall.asp
Last accessed: 27/04/2004

[eBay 2005]
eBay Inc. (2005). Reputation — eBay Feedback: Overview. *Webpage*.
Available at: http://pages.ebay.co.uk/help/buy/reputation-ov.html
Last accessed: 20/12/2005

[EPIC, Junkbusters 2000]
EPIC and Junkbusters (2000). Pretty Poor Privacy: An Assessment of P3P and Internet Privacy. *Webpage*.
Available at: http://www.epic.org/Reports/prettypoorprivacy.html
Last accessed: 08/10/2001

[European Commission 2002]
European Commission (2002). Questionnaire on the implementation of the Data Protection Directive (95/46/EC). Your voice in Europe. *Webpage*.
Available at: http://europa.eu.int/yourvoice/results/204/index_en.htm
Last accessed: 18/06/2003

[Gellersen, Schmidt et al. 2002]
Gellersen, H.-W., A. Schmidt, et al. (2002). Multi-Sensor Context-Awareness in Mobile Devices and Smart Artifacts. *Mobile Networks and Applications* **7**(5): pp. 341-351.
Available at: http://www.smart-its.org/publication/sensors-in-mobile-devices.monet.pdf
Last accessed: 14/12/2005

[Goecks, Mynatt 2002]
Goecks, J. and E. Mynatt (2002). Enabling privacy management in ubiquitous computing environments through trust and reputation systems. *Workshop on Privacy in Digital Environments: Empowering Users*. CSCW 2002. New Orleans, LA USA.
Available at: http://www.cc.gatech.edu/~everyday-computing/projects/saori/pubs/ReputationTrust-cscw2002.pdf
Last accessed: 19/12/2005

[Greenhalgh 2002]
Greenhalgh, C. (2002). EQUIP: a Software Platform for Distributed Interactive Systems. *Equator Technical Report 02-002*, University of Nottingham.
Available at: http://citeseer.ist.psu.edu/539227.html
Last accessed: 26/04/2007

[Grimm, Rossnagel 2000]
Grimm, R. and A. Rossnagel (2000). Can P3P Help to Protect Privacy WorldWide? *ACM Multimedia Workshop*, Mariana Del Rey, CA, USA, ACM Press.

[Guardian 2002]
Guardian (2002). ICM Poll. *The Guardian* Big Brother **2002**(07/09/2002):pp. 3.

[GVU's WWW Surveying Team 1998]
GVU's WWW Surveying Team (1998). GVU's 10th WWW User Survey, GVU Center, College of Computing Georgia Institute of Technology. *Webpage*.
Available at: http://www.gvu.gatech.edu/user_surveys/survey-1998-10/
Last accessed: 08/12/2005

[Harter, Hopper 1994]
Harter, A. and A. Hopper (1994). A Distributed Location System for the Active Office. *IEEE Network* **8**(1).
Available at: http://www-lce.eng.cam.ac.uk/publications/files/tr.94.1.pdf
Last accessed: 23/06/2004

[Hewlett-Packard Company 2003]
Hewlett-Packard Company (2003). User's Guide hp iPAQ Pocket PC h4000 Series. Document Part Number: 343434-001. *Webpage*.
Available at:
http://h20000.www2.hp.com/bc/docs/support/SupportManual/c00046424/c00046424.pdf
Last accessed: 14/12/2005

[Hill, Culler 2002]
Hill, J. L. and D. E. Culler (2002). Mica: a wireless platform for deeply embedded networks. *IEEE Micro*. **22:** pp. 12-24.
Available at: http://www.cs.berkeley.edu/~culler/cs294-f03/papers/micaarch.pdf
Last accessed: 14/12/2005

[Hong, Landay 2001]
Hong, J. I. and J. A. Landay (2001). An Infrastructure Approach to Context-Aware Computing. *Human-Computer Interaction* **16**(2): pp. 287-303.

[Indulska, Robinson et al. 2003]
Indulska, J., R. Robinson, et al. (2003). Experiences in Using CC/PP in Context-Aware Systems. *4th International Conference on Mobile Data Management*, Melbourne, Australia.
Available at: http://citeseer.ist.psu.edu/561612.html
Last accessed: 26/02/2006

[Intille, Larson et al. 2005]
Intille, S. S., K. Larson, et al. (2005). The PlaceLab: a live-in laboratory for pervasive computing research. *Proceedings of Pervasive 2005 Video Program*
Available at:
http://www.pervasive.ifi.lmu.de/adjunct-proceedings/video/p183-186.pdf
http://www.media.mit.edu/%7Eintille/videos/Pervasive05DIVX.avi
Last accessed: 13/12/2005

[IBM alphaWorks]
IBM alphaWorks (2005). P3P Policy Editor (Beta 1.11). *Webpage*.
Available at: http://www.alphaworks.ibm.com/tech/p3peditor
Last accessed: 28/02/2003

[Java Community Process 2001]
Java Community Process (2001). JSR 53: JavaTM Servlet 2.3 and JavaServer PagesTM 1.2 Specifications (Final Release). *Webpage*.
Available at: http://jcp.org/en/jsr/detail?id=53
Last accessed: 21/12/2005

[Java Community Process 2002A]
Java Community Process (2002). JSR 36: J2ME Connected Device Configuration 1.0a Maintenance Release (Final Release updated on August 12, 2002). *Webpage*.
Available at: http://www.jcp.org/en/jsr/detail?id=36
Last accessed: 21/12/2005

[Java Community Process 2002B]
Java Community Process (2002). JSR 46: J2ME Foundation Profile 1.0a Maintenance Release (Final Release updated on August 12, 2002). *Webpage*.
Available at: http://www.jcp.org/en/jsr/detail?id=46
Last accessed: 21/12/2005

[Java Community Process 2002C]
Java Community Process (2002). JSR 62: J2ME Personal Profile Specification 1.0 (Final Release). *Webpage*.
Available at: http://www.jcp.org/en/jsr/detail?id=62
Last accessed: 21/12/2005

[Java Community Process 2003]
Java Community Process (2003). JSR-000179 Location API for J2ME. *Webpage*.
Available at: http://jcp.org/aboutJava/communityprocess/final/jsr179/index.html
Last accessed: 08/12/2005

[JRC P3P Resource Centre 2005A]
JRC P3P Resource Centre (2005). JRC P3P Proxy Version 2.0. *Webpage*.
Available at: http://p3p.jrc.it/
Last accessed: 08/12/2005

[JRC P3P Resource Centre 2005B]
JRC P3P Resource Centre (2005). JRC Ruleset editor (1.3 Beta). *Webpage*.
Available at: http://p3p.jrc.it/
Last accessed: 28/02/2003

[Kaliski 2003]
Kaliski, B. (2003). TWIRL and RSA Key Size. RSA Laboratories. *Webpage*.
Available at: http://www.rsasecurity.com/rsalabs/node.asp?id=2004
Last accessed: 27/12/2005

[Kidd, Orr et al. 1999]
Kidd, C. D., R. Orr, et al. (1999). The Aware Home: A Living Laboratory for Ubiquitous Computing Research. *Second International Workshop on Cooperative Buildings - CoBuild'99*, Pittsburgh, USA.
Available at:
http://www.cc.gatech.edu/fce/ahri/publications/cobuild99_final.PDF
Last accessed: 26/02/2004

[Kindberg, Barton et al. 2002]
Kindberg, T., J. Barton, et al. (2002). People, Places, Things: Web Presence for the Real World. *Mobile Networks and Applications* **7**(5): pp. 365-376.
Available at: http://www.hpl.hp.com/techreports/2001/HPL-2001-279.pdf
Last accessed: 13/12/2005

[Korpipää, Mäntyjärvi 2003]
Korpipää, P. and J. Mäntyjärvi (2003). An Ontology for Mobile Device Sensor-Based Context. *Context 2003*, Stanford, Califonia, USA, Springer-Verlag Berlin Heidelberg. LNAI, pp. 451-458.

[Korpipää, Mäntyjärvi et al. 2003]
Korpipää, P., J. Mäntyjärvi, et al. (2003). Managing Context Information in Mobile Devices. *IEEE Pervasive Computing*. **2:** pp. 43-51.

[Kruchten 2003]
Kruchten, P. (2003). *The Rational Unified Process: An Introduction*, Addison-Wesley Professional.

[Laerhoven, Schmidt et al. 2002]
Laerhoven, K. V., A. Schmidt, et al. (2002). Pin&Play: Networking Objects through Pins. *4th International Conference on Ubiquitous Computing, UbiComp 2002*, Göteborg, Sweden, Springer-Verlag Berlin Heidelberg.

[Landesberg, Levin et al. 1998]
Landesberg, M. K., T. M. Levin, et al. (1998). Privacy Online: A Report to Congress, Federal Trade Commission.
Available at: http://www.ftc.gov/reports/privacy3/priv-23a.pdf
Last accessed: 19/12/2005

[Langheinrich 2001]
Langheinrich, M. (2001). Privacy by Design - Principles of Privacy-Aware Ubiquitous Systems. *UbiComp 2001*, Atlanta, Georgia, USA, Springer-Verlag Berlin Heidelberg.

[Langheinrich 2002]
Langheinrich, M. (2002). A Privacy Awareness System for Ubiquitous Computing Environments. *4th International Conference on Ubiquitous Computing, UbiComp 2002*, Göteborg, Sweden, Springer-Verlag Berlin Heidelberg.

[Larousse 1994]
Larousse plc. (1994). The Chambers Dictionary. *Microsoft Bookshelf - British Reference Collection*.

[Lau, Etzioni et al. 1999]
Lau, T., O. Etzioni, et al. (1999). Privacy Interfaces for Information Management. *Communications of the ACM* **42**(10): pp. 88-94.
Available at: ftp://ftp.cs.washington.edu/tr/1998/02/UW-CSE-98-02-01.PS.Z
Last accessed: 07/01/2006

[Lederer, Mankoff et al. 2003]
Lederer, S., J. Mankoff, et al. (2003) Who Wants to Know What When? Privacy Preference Determinants in Ubiquitous Computing. *Extended Abstracts of CHI 2003, ACM Conference on Human Factors in Computing Systems*.
Available at: http://www.cs.berkeley.edu/projects/io/publications/privacy-chi03.pdf
Last accessed: 07/01/2006

[Lederer, Hong et al. 2003]
Lederer, S., Hong, J.I., Jiang, X., Dey, A.K., Landay, J.A., Mankoff, J. Towards Everyday Privacy for Ubiquitous Computing. *Technical Report UCB-CSD-03-1283*, Computer Science Division, University of California, Berkeley.
Available at: http://www.cs.berkeley.edu/projects/io/publications/privacy-techreport03a.pdf
Last accessed: 07/01/2006

[Loomis 2001]
Loomis, D. (2001). The TINI Specification and Developer's Guide, *Addison Wesley Professional*.
Available at: http://www.maxim-ic.com/products/tini/pdfs/tinispec.pdf
Last accessed: 21/12/2005

[Mascone 2002]
Masone, C. (2002). Role Definition Language (RDL): A language to describe context-aware roles. *Dartmouth Computer Science Technical Report*. TR2002-426.
Available at: ftp://ftp.cs.dartmouth.edu/TR/TR2002-426.pdf
Last accessed: 05/10/2004

[McCarthy 1998]
McCarthy, J. F. (1998). MusicFX: An Arbiter of Group Preferences. *AAAI Spring Symposium on Intelligent Environments*, Palo Alto.
Available at: http://citeseer.ist.psu.edu/288814.html
Last accessed: 03/12/2005

[Megginson 2001]
Megginson, D. (2001). RDF Filter. Version 1.0 alpha. *Webpage*.
Available at: http://sourceforge.net/projects/rdf-filter/
Last accessed: 20/12/2005

[Minami, Kotz 2002]
Minami, K. and D. Kotz (2002). Controlling access to pervasive information in the "Solar" system. *Dartmouth Computer Science Technical Report TR2002-422*.
Available at: ftp://ftp.cs.dartmouth.edu/TR/TR2002-422.pdf
Last accessed: 19/12/2005

[Mobil Turism 2005]
Mobil Turism (2005). Mobil Guide. *Webpage*.
Available at: http://www.mobilturism.se/
Last accessed: 03/12/2005

[National Statistics 2005]
National Statistics (2005). Individuals accessing the Internet – National Statistics Omnibus Survey. Webpage. (Published 26/04/2005)
Available at: http://www.statistics.gov.uk/cci/nugget.asp?id=8
Last accessed: 11/08/2005

[Nilsson, Lindskog et al 2001]
Nilsson, M., H. Lindskog, et al. (2001). Privacy Enhancement in the Mobile Internet. *Security and Control of IT in Society-II, IFIP SCITS-II*, Bratislava, Slovakia.

[NIST 2001]
National Institute of Standards and Technology (2001). Announcing the ADVANCED ENCRYPTION STANDARD (AES). *Webpage*.
Available at: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
Last accessed: 21/12/2005

[Osbakk 2006]
Osbakk P. (2006). How to get started with iButtons using an iPAQ and the Jeode VM. *Webpage*.
Available at: http://www.osbakk.com/pub/iButton_Instructions.pdf
Last accessed: 20/04/2006

[Osbakk, Ryan 2002]
Osbakk, P. and N. Ryan (2002). Context, CC/PP, and P3P. *UbiComp 2002 Adjunct Proceedings*, Göteborg, Sweden, Viktoria Institute. pp. 9-10.
Available at:
http://www.viktoria.se/ubicomp/ubicomp_adjunct_proceedings.pdf
Last accessed: 20/12/2005

[Osbakk, Ryan 2003]
Osbakk, P. and N. Ryan (2003). A Privacy Enhancing Infrastructure for Context-Awareness. *1st UK-UbiNet Workshop*, London, UK.
Available at: http://www-dse.doc.ic.ac.uk/Projects/UbiNet/ws2003/papers/osbakk.pdf
Last accessed: 19/12/2005

[Osbakk, Ryan 2004A]
Osbakk, P. and N. Ryan (2004). Expressing Privacy Preferences in terms of Invasiveness. *2nd UK-UbiNet Workshop*, Cambridge, UK.
Available at: http://www-dse.doc.ic.ac.uk/Projects/UbiNet/ws2004/Papers/18-Osbakk.pdf
Last accessed: 05/01/2006

[Osbakk, Ryan 2004B]
Osbakk, P. and N. Ryan (2004). The development of a privacy-enhancing infrastructure: Some interesting findings. *Ubicomp Privacy: Current Status and Future Directions Workshop, UbiComp 2004*, Nottingham, UK.
Available at: http://www.cs.kent.ac.uk/pubs/2004/1977/content.pdf
Last accessed: 27/12/2005

[Osbakk, Rydgren 2005]
Osbakk, P. and E. Rydgren (2005). Ubiquitous Computing for the Public. *Pervasive Mobile Interaction Devices Workshop*, Pervasive 2005, Munich, Germany.
Available at:
http://www.medien.ifi.lmu.de/permid2005/pdf/PatrikOsbakk_Permid2005.pdf
Last accessed: 13/12/2005

[Pascoe, Morse et al. 1998]
Pascoe, J., D. Morse, et al. (1998). Developing Personal Technology for the Field. *Personal Technologies* 2(1): pp. 28-36.

[P3P Builder 2005]
P3PBuilder (2005). P3PBuilder. *Webpage*.
Available at: http://www.p3pbuilder.com/
Last accessed: 27/12/2005

[P3P Writer 2005]
P3PWriter (2005). P3PWriter. *Webpage*.
Available at: http://www.p3pwriter.com/
Last accessed: 27/12/2005

[Rivest, Shamir et al. 1977]
Rivest, R., A. , A. Shamir, et al. (1977). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM* **21**(2): pp. 120-126.
Available at: http://theory.lcs.mit.edu/~rivest/rsapaper.pdf
Last accessed: 21/12/2005

[Rudström, Svensson et al. 2004]
Rudström, Å., M. Svensson, et al. (2004). MobiTip: Using Bluetooth as a Mediator of Social Context. *Ubicomp 2004 Adjunct Proceedings (demo)*, Nottingham, UK.
Available at: http://ubicomp.org/ubicomp2004/adjunct/demos/rudstrom.pdf
Last accessed: 13/12/2005

[Russo, Sukojo 2004]
Russo, J., A. Sukojo, et al. (2004). SmartWave –Intelligent Meal Preparation System to Help Older People Live Independently. *Second International Conference On Smart homes and health Telematic (ICOST2004)*, Singapore.
Available at: http://www.harris.cise.ufl.edu/projects/publications/Russo-ICOST2004-SmartWave-final.pdf
Last accessed: 14/12/2005

[Ryan, Pascoe et al. 1997]
Ryan, N., J. Pascoe, et al. (1997). Enhanced Reality Fieldwork: the Context-aware Archaeological Assistant. *Computer Applications in Archaeology 1997*, Tempus Reparatum.
Available at: http://www.cs.kent.ac.uk/pubs/1998/616/content.html
Last accessed: 13/12/2005

[Ryan, Pascoe et al. 1998]
Ryan, N., J. Pascoe, et al. (1998). FieldNote: extending a GIS into the field. *New Techniques for Old Times: Computer Applications in Archaeology, 1998*, Archaeopress, Oxford, UK.
Available at: http://www.cs.kent.ac.uk/pubs/1999/1015/content.html
Last accessed: 05/10/2004

[Ryan, Pascoe et al. 1999]
Ryan, N., J. Pascoe, et al. (1999). FieldNote: a handheld information system for the field. *TeleGeo'99, 1st International Workshop on TeleGeoProcessing*, Lyon, France.

[Ryan 2005]
Ryan, N. (2005). Smart Environments for Cultural Heritage. *Reading the Historical Spatial Information in the World. 24th International Symposium*, Kyoto, Japan.
Available at: http://www.cs.kent.ac.uk/pubs/2005/2053/content.pdf
Last accessed: 13/12/2005

[Salber, Dey et al. 1999]
Salber, D., A. K. Dey, et al. (1999). The Context Toolkit: Aiding the Development of Context-Enabled Applications. *Proceedings of CHI99, ACM SIGCHI Conference on Human Factors in Computing Systems*, Pittsburgh, PA, US.
Available at: http://www.cc.gatech.edu/fce/contexttoolkit/pubs/chi99.pdf
Last accessed: 21/06/2004

[Sandhu, Coyne et al. 1996]
Sandhu, R. S., E. J. Coyne, et al. (1996). Role-Based Access Control Models. Revised October 26, 1995. *IEEE Computer* **29**(2): pp. 38-47.
Available at: http://csrc.nist.gov/rbac/sandhu96.pdf
Last accessed: 20/12/2005

[Scheemaecker 2003]
Scheemaecker, M. D. (2003). NanoXML. Version 2.2.3. *Webpage*.
Available at: http://nanoxml.cyberelf.be/index.html
Last accessed: 20/12/2005

[Schilit, Adams et al. 1994]
Schilit, B. N., N. Adams, et al. (1994). Context-Aware Computing Applications. *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, USA.
Available at: http://citeseer.ist.psu.edu/schilit94contextaware.html
Last accessed: 13/12/2005

[Schilit 1995]
Schilit, B. N. (1995). A Context-Aware System Architecture for Mobile Distributed Computing. *Columbia University*.
Available at: http://seattleweb.intel-research.net/people/schilit/schilit-thesis.pdf
Last accessed: 08/12/2005

[Schilit, Theimer 1994]
Schilit, B. N. and M. Theimer (1994). Disseminating Active Map Information to Mobile Hosts. *IEEE Network* **8**(5): pp. 22-32.
Available at: http://seattleweb.intel-research.net/people/schilit/ams.pdf
Last accessed: 05/10/2004

[Sony Ericsson 2004]
Sony Ericsson (2004). P910 Series White Paper. *Webpage*.
Available at: http://developer.sonyericsson.com/getDocument.do?docId=66933
Last accessed: 14/12/2005

[Stajano 2002]
Stajano, F. (2002). Security for Ubiquitous Computing, *John Wiley & Sons*.

[Stationery Office 1998]
Stationery Office. (1998). Data Protection Act, 1998 (Public General Acts - Elizabeth II). *The Stationery Office Books*.
Available at: http://www.opsi.gov.uk/acts/acts1998/19980029.htm
Last accessed: 20/12/2005

[Sun Microsystems 1998]
Sun Microsystems Inc. (1998). PersonalJava API Specification. Version 1.1. *Webpage*.
Available at: http://java.sun.com/products/personaljava/spec-1-1/pJavaSpec.html
Last accessed: 21/12/2005

[Sun Microsystems 2002]
Sun Microsystems Inc. (2002). Datasheet Java 2 Platform, Micro Edition. *Webpage*.
Available at: http://java.sun.com/j2me/j2me-ds.pdf
Last accessed: 21/12/2005

[Telia 2005]
Telia (2005). FriendFinder. *Webpage*.
Available at: http://friendfinder.telia.se/
Last accessed: 03/12/2005

[Toshiba 2005]
Toshiba (2005). Tecra M4. *Webpage*.
Available at: http://uk.computers.toshiba-europe.com/cgi-bin/ToshibaCSG/jsp/SUPPORTSECTION/discontinuedProductPage.do?PRODUCT_ID=104798&DISC_MODEL=1&service=UK
Last accessed: 11/07/2007

[Wakefield 2005]
Wakefield, J. (2005). UK laws are failing to deter spam. BBC News. *Webpage*.
Available at: http://news.bbc.co.uk/1/hi/technology/4466053.stm
Last accessed: 27/12/2005

[Want, Hopper 1992]
Want, R. and A. Hopper (1992). Active Badges and Personal Interactive Computing Objects. *IEEE Transactions of Consumer Electronics* **38**(1): pp. 10-20.
Available at: http://www-lce.eng.cam.ac.uk/publications/files/tr.2001.8.pdf
Last accessed: 23/06/2004

[Want, Hopper 1992B]
Want, R., A. Hopper, et al. (1992). The Active Badge Location System. *ACM Transactions on Information Systems* **10**(1): pp. 91-102.
Available at: http://sandbox.xerox.com/want/papers/ab-tois-jan92.pdf
Lastr accessed: 13/12/2005

[Ward 2005]
Ward, M. (2005). Virus flood threatens home users. BBC News. *Webpage*.
Available at: http://news.bbc.co.uk/1/hi/technology/4080420.stm
Last accessed: 27/12/2005

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

[Weiser, Gold et al.1999]
Weiser, M., R. Gold, et al. (1999). The origins of ubiquitous computing research at PARC in the late 1980s. *IBM Systems Journal* **38**(4): pp. 693-696.
Available at: http://www.research.ibm.com/journal/sj/384/weiser.pdf
Last accessed: 13/03/2003

[Weiser 2002]
Weiser, M. (2002). The Computer for the 21st Century. *Pervasive Computing*. **1**(1): pp. 19-25. Reprint.

[Westin 1970]
Westin, A. F. (1970). Privacy and freedom. London, *Bodley Head*.

[Wilson 2001A]
John Wilson (2001). MinML2 a namespace aware minimal XML parser. Version 0.3. The Wilson Partnership. *Webpage*.
Available at: http://www.wilson.co.uk/xml/minml2.htm
Last accessed: 20/12/2005

[Wilson 2001B]
John Wilson (2001). MinML a minimal XML parser. Version 1.7. The Wilson Partnership. *Webpage*.
Available at: http://www.wilson.co.uk/xml/minml.htm
Last accessed: 20/12/2005

[World Wide Web Consortium 2002A]
World Wide Web Consortium (2002). The Platform for Privacy Preferences 1.0 (P3P1.0) Specification [W3C Recommendation 16 April 2002]. *Webpage*.
Available at: http://www.w3.org/TR/2002/REC-P3P-20020416/
Last accessed: 03/12/2005

[World Wide Web Consortium 2002B]
World Wide Web Consortium (2002). A P3P Preference Exchange Language 1.0 (APPEL1.0) [W3C Working Draft 15 April 2002]. *Webpage*.
Available at: http://www.w3.org/TR/2002/WD-P3P-preferences-20020415/
Last accessed: 03/12/2005

[World Wide Web Consortium 2004A]
World Wide Web Consortium (2004). Resource Description Framework (RDF): Concepts and Abstract Syntax [W3C Recommendation 10 February 2004]. *Webpage*.
Available at: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/
Last accessed: 19/12/2005

[World Wide Web Consortium 2004B]
World Wide Web Consortium (2004). RDF Primer [W3C Recommendation 10 February 2004]. *Webpage*.
Available at: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/
Last accessed: 19/12/2005

[World Wide Web Consortium 2004C]
World Wide Web Consortium (2004). RDF/XML Syntax Specification (Revised) [W3C Recommendation 10 February 2004]. *Webpage*.
Available at: http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/
Last accessed: 19/12/2005

[World Wide Web Consortium 2004D]
World Wide Web Consortium (2004). RDF Vocabulary Description Language 1.0: RDF Schema [W3C Recommendation 10 February 2004]. *Webpage*.
Available at: http://www.w3.org/TR/2004/REC-rdf-schema-20040210/
Last accessed: 19/12/2005

[World Wide Web Consortium 2004E]
World Wide Web Consortium (2004). Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0 [W3C Recommendation 15 January 2004]. *Webpage*.
Available at: http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/
Last accessed: 20/12/2005

[World Wide Web Consortium 2005A]
World Wide Web Consortium (2005). RDF Validation Service. *Webpage*.
Available at: http://www.w3.org/RDF/Validator/
Last accessed: 19/12/2005

[World Wide Web Consortium 2005B]
World Wide Web Consortium (2005). References for P3P Implementations. *Webpage*.
Available at: http://www.w3.org/P3P/implementations
Last accessed: 08/08/2005

[Zhang, Parashar 2004]
Zhang, G. and M. Parashar (2004). Context-aware Dynamic Access Control for Pervasive Applications. *Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2004)*, San Diego, CA, USA.
Avilable at: http://www.caip.rutgers.edu/TASSL/Papers/automate-sesame-cnds-04.pdf
Last accessed: 26/02/2004