AmbieSense – A System and Reference Architecture for Personalised Context-Sensitive Information Services for Mobile Users

Hans Myrhaug[1], Nik Whitehead[2], Ayse Goker[2], Tor Erlend Faegri[3] and Till Christopher Lech[3]

[1] SINTEF Information and Communication Technology, N-7465 Trondheim, Norway
toref, hansim@sintef.no
[2] School of Computing, The Robert Gordon University, Aberdeen AB25 1HG, Scotland
njw, asg@comp.rgu.ac.uk
[3]CognIT AS, Metzers gt. 4, Oslo, Norway
till@cognit.no
http://www.ambiesense.net/

Abstract. The purpose of AmbieSense is to provide personalised, context-sensitive information to the mobile user. It is about augmenting digital information to physical objects, rooms, and areas. The aim is to provide relevant information to the right user and situation. Digital content is distributed from the surroundings and onto your mobile phone. An ambient information environment is provided by a combination of context tag technology, a software platform to manage and deliver the information, and personal computing devices to which the information is served. This paper describes how the AmbieSense reference architecture has been defined and used in order to deliver information to the mobile citizen at the right time, place and situation. Information is provided via specialist content providers. The application area addresses the information needs of travellers and tourists.

1 Introduction to AmbieSense

AmbieSense addresses ambient, personalised, and context-sensitive information systems for mobile users. The overall goal of such systems is to help achieve the digital, ambient environments that make user's information-related tasks easier by adapting to user's context and personal requirements. Our approach to solve this is illustrated in Figure 1 below. The figure illustrates the AmbieSense reference architecture at an overall level. It can be used to build various digital information channels for mobile users. The objective is to provide the correct information to the right situation of the mobile user. The figure depicts three central cornerstones of the system: *Content Service Providers (CSP)*, *context tags* and *mobile/travelling users*.

Information or *content* is provided by Content Service Providers, offering net-based, digital information services to their customers. This is currently achieved by direct communication between the information consumers (i.e. the mobile users) and the CSP. A key objective of CSPs is to increase the value of their services by

increasing the reach, relevance, and accuracy of information provided to the consumer.

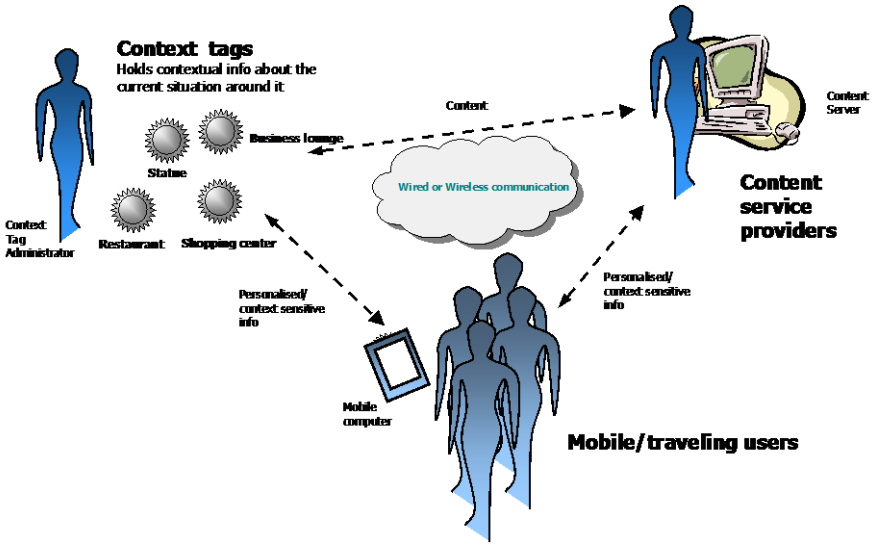

Figure 1: The AmbieSense overall reference architecture

AmbieSense has designed and implemented *context tags* (see Figure 2) as part of the project. Context tags are miniaturized, wireless computers placed at strategic points in the user's environment that can relay content from the CSP, prioritised with context information, to the mobile device. The context tags have computing capabilities which enable different software applications to run on them. Context tags differ from other disappearing computers because one can exploit contextual information about both the tag environment and the user in range to provide relevant content. We have designed a product family of context tags to fulfil different application needs within ambient computing. The tags can differ with respect to storage capacity, network communication (i.e. Bluetooth, WLAN, and Ethernet), computing speed, and programming possibilities. They are based on Linux operating system, and can have none or several of the AmbieSense software

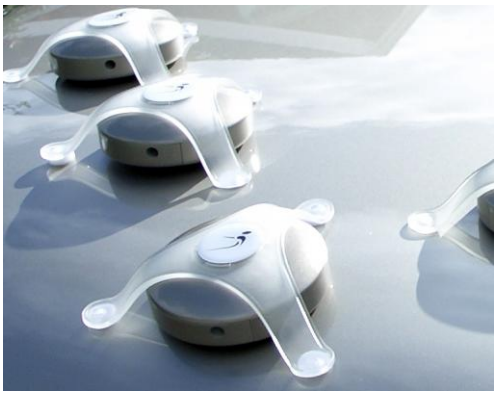

Figure 2: The AmbieSense context tags. They communicate via Bluetooth to handheld devices nearby, and are 12 cm in diameter. (Hardware and design by SINTEF ICT)

components running on them –depending on the application needs (see Figure 3 and 4).

Mobile users are the consumers of information services provided by the system. We assume that they use some kind of mobile computer, e.g. a PDA or a smart phone to interact with these services. A central idea is to associate information with objects in the surroundings. This can be through seamless access to content when people are near tags or by creating an environment that stimulates the user's curiosity and encourages him to look for information in the surroundings. AmbieSense applications can identify content for an individual by exploiting contextual knowledge about the user.

In summary, AmbieSense seeks to address the requirements of ambient content services provision and mobile users by improving the reach, accuracy and timeliness of content delivered to the mobile user. Each application can have different system architectures instantiated from the AmbieSense reference architecture. The next sections explain how this is achieved.

1.1 Infrastructure and Framework

The AmbieSense framework can run on different infrastructures that enable mobile users to access digital content.

Infrastructure. AmbieSense can run on a range of infrastructure technologies, including wireless communications, hand-held computers, PDAs, smart phones, information and application servers. Each technology has been targeted because they can provide important, value-adding functionality to the framework in terms of integration with both new and existing systems.

AmbieSense Framework. One of the main technical outcomes of the project, are the context tags (hardware) together with the technical platform (software). The tags and the technical platform are both meant to be key mechanisms for ambient content services and applications, with the purpose to build different applications. The main architectural components of the AmbieSense Framework are depicted in Figure 3 and 4. For instance, the underlying Content Integration Platform (CIP) provides the ambient content management functionality to an application. It enables ambient access to content from mobile devices with limited storage. One important part of the CIP is a search engine that can run on mobile phones, context tags, and on content servers. Another component is the context middleware, which supports the context-aware applications. It enables applications to store, update, and retrieve contexts and their structural templates (in the project, we have had our own context structures, but the middleware supports the development and use of others too).

2 The AmbieSense Reference Architecture

The AmbieSense reference architecture is documented using principles from the RM-ODP (Reference Model - Open Distributed Processing) method [1]. This method

was selected mainly because it is an established standard. It comes with five predefined viewpoints: enterprise, information, computational, engineering, and technological viewpoints. Each viewpoint helps to visualise the range of different aspects that need to be addressed when constructing architectures.

In this paper, we describe the following three viewpoints[1]:

1. *Enterprise viewpoint – focusing on the overall functionality of the system towards its users and stakeholders, described in terms of user roles and actors.*
2. *Computational viewpoint – focusing on the high-level component descriptions and interactions between these components.* The computational viewpoint is documented using diagrams showing the functional decomposition into components and their interfaces.
3. *Engineering viewpoint – focusing on how the concrete configurations of the reference architecture can be deployed to real systems.* The engineering viewpoint is documented using deployment diagrams showing communication and distribution aspects of system components. It also addresses performance and capacity issues by suggesting how these quality requirements can be satisfied by the different configurations.

2.1 Enterprise Viewpoint

Content: Content is the information that flows in the AmbieSense system. From the user's point of view, content is any data that the user can receive, either manually by requesting for it, or automatically in respect of some settings. Usually, content is temporarily stored in the form of audio, movie, image, text, or mark-up files before it is presented for the user.

Context: Context can be defined as a description of the aspects of a situation [2]. The period of a context can range from being a very short moment to many years. The current context can depend on several criteria, e.g. the location, mental state, etc. The user's current context can have a direct influence on the functionality of the user application (e.g. the application can present relevant information to the user or choose not to present some information based on the current context). One may speak of several types of contexts, depending on your application and your needs for info – and of your view of what info/data is important to capture/describe a situation.

In AmbieSense, context technology is a mechanism that can capture contexts structures, and links between contexts and content. However, we argue that there should be some common structure for user contexts, which is easy to reuse across domains (such as different applications). What makes domains differ is mainly that the relevance and importance of attributes within the context structure differ. Redundant attributes may exist in the context as their relevance can change over time.

[1] Note that the *technological viewpoint* is not addressed by the AmbieSense reference architecture, because it is platform and implementation neutral. Technological aspects should be addressed at the time of derivation of a specific system, when detailed technological requirements become available. Also note that the *information viewpoint* exists in technical project reports and is too large to be covered within the scope of this paper.

Users: The users are the consumers of content that a CSP provides. They may also indirectly be consumers of contexts. A user will be able to receive information according to the current context.

Context Tags: AmbieSense has developed new, miniaturized, wireless computers called *context tags*. A context tag is an entity that enables the binding of a location to a context (or a set of contexts), and content. The context tag is realised as an embedded computer with a Bluetooth interface for communication. The communication facility is used for the exchange of software, content and context.

In its simplest form, a context tag only gives a reference to a location to be used by a mobile user. More advanced context tags enable other services. For instance, a context tag can have a web server, a search engine, and other software installed, available for mobile devices nearby to use.

2.5 Computational Viewpoint

The computational viewpoint is concerned with the functional decomposition of the system into components and the interaction patterns between the components (services) of the system, described through their interfaces.

Overview. The AmbieSense Reference Architecture consists of a set of main components, described below following a top-down sequence. Figure 3 illustrates the layers and the organisation of the architecture pattern. The light grey components are part of the AmbieSense Framework; the darker grey components are part of each application/solution developed on top of the framework.

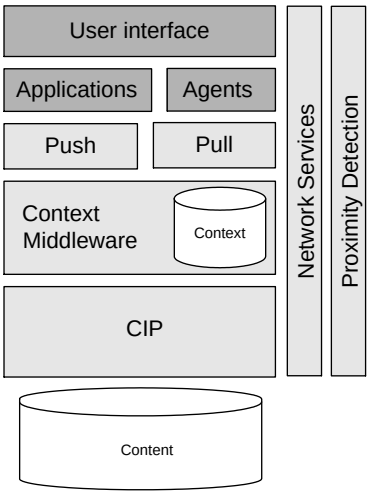

Figure 3: Functional decomposition of AmbieSense architecture

By layering, we assume that components form a stack that prescribes how components interact with each other. For example, in Fig.2 the user interface is illustrated on top of applications and agents. This implies that the user interface uses the services offered by applications and agents. Likewise, applications and agents use the services from the push and pull components. In some system architectures, one or more of the indicated layers may not be present, thus the reference architecture allows interaction between non-adjacent layers. However, if the layer is present, then the layering principle should be adhered to.

- *User Interface*. The user interface enables the user's interaction with the system. The design of the user interface is based upon the requirements for the particular application. The user interface uses the services offered by the developed application and/or agents

- *Applications and Agents*. Applications and agents[2] are developed according to the needs of each specific solution. Developers may choose whether to exploit agents to provide a solution. Applications (and agents) use the services of the layers below, primarily the push and pull services, but also services from the context middleware and CIP when appropriate.
- The *Push* and *Pull* components implement functionality for content distribution, supporting the two different principles push and pull. The Push and Pull components use the context management functionality of the context middleware together with the content provisioning capabilities of the CIP to enable context-aware content distribution.
- *Context Middleware*. The context middleware is responsible for context management (context-storage, -retrieval, and -matching functions). The context middleware supports additional functionality such as linking content to contexts.
- *CIP*. The CIP (Content Integration Platform) is a composite service component that deals with content management in terms of capturing, inclusion, integration, and distribution of content to users. It adds functionality for personalisation and customisation – all within an integrated platform. The CIP provides a single interface to heterogeneous content bases underneath while adding useful functionality for caching, and application protocols in an integrated environment. CIP Light is a minimized version of the CIP designed for mobile devices. A mobile device does not need the same functionality as does a CSP server, and the distinction between CIP and CIP Light reflects this.
- *Network Services*. Because AmbieSense is inherently distributed, networking capabilities are used between the platforms (i.e. mobile computer, context tag, and CSP platform). Networking capabilities are provided by industry standard technologies, such as Bluetooth, WLAN, and GPRS.
- *Proximity Detection*. In addition, a subcomponent called proximity detection will enable detection of mobile computers in the vicinity of the context tag.

2.6 Engineering Viewpoint

The engineering viewpoint is concerned with the design of distribution-oriented aspects, i.e. how components are physically deployed to the different machines and infrastructure required to support distribution. The engineering viewpoint specifies a networked computing infrastructure that supports the system structure defined in the computational specification and provides the distribution transparencies that it defines.

It is important to note that although we discuss the context tag in all configurations, it is possible to implement the AmbieSense Reference Architecture without them. They might be omitted or other style computers with similar capabilities can be used

[2] In multi-agent systems, agents are programs that act in a self-interested manner in their dealings with numerous other agents inside a computer. This arrangement can mimic almost any interactive system: a stock market; a habitat; even a business supply-chain.

instead. For example, wireless access points and other embedded computers can be used instead of the tags. However, they should be re-programmable for the application developer. Hence, a context tag is an open hardware platform.

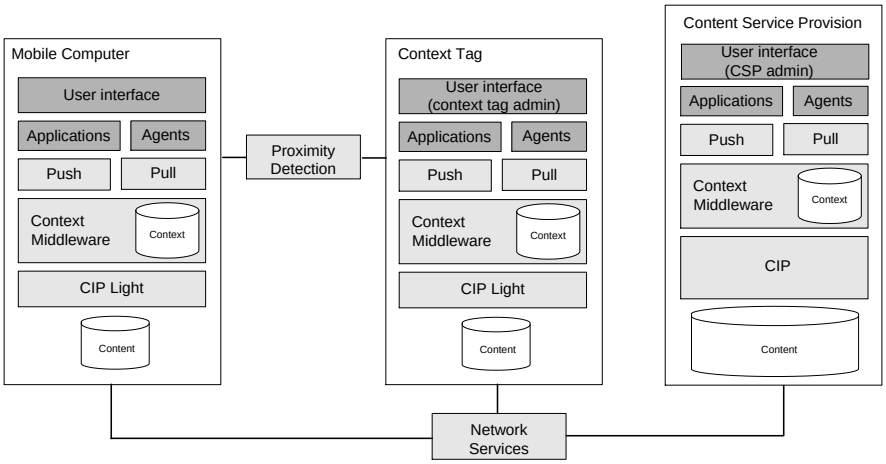

Figure 4: Reference Architecture Platform

Figure 4 illustrates how the various functional components are organised logically in the reference architecture. It does *not* show a concrete implementation of it. Rather, it shows how each of the three platforms (mobile computer, context tag and content service provision platform) is able to function as nearly independent execution environments. Now this is not, however, a very likely scenario in practice. It is merely a demonstration of the flexibility of the reference architecture to support widely different implementations dictated by the application and user requirements. To illustrate this, three examples of concrete system architectures based upon the reference architecture are found below (see Figure 5 for deployment diagrams of the functional components).

Example 1: Thin client, rich Context Tag configuration. Certain solutions require low complexity mobile computer configurations. For example, if a solution was going to support the use of smart phones with very limited storage and processing capacities, most of the processing should be allocated to the Context Tag or the Content Service Provision platforms (see Example 1 in Figure 5).

Example 2: Medium-weight Mobile Computer, rich Context Tag configuration. In scenarios where more generic and sophisticated context processing such as context matching is required, the Context Middleware can be deployed also to the Mobile Computer. Naturally, this assumes more processing power and storage available on the mobile device. Example 2 in Figure 5 illustrates the deployment scenario. In this example the application can pull content from the context tag. The pull can be based upon the current context on the mobile computer, and the content related to it or other similar contexts, may be delivered from the tag and to the user.

Example 3: Rich Mobile Computer, medium weight Context Tag, medium weight Content Service Provision platform. The previous configurations have not

included the Content Service Provision platform. For many technical solutions, the integration of content from existing content management systems is critical. This may be a good reason to consider configurations that include a content server that can provide the link to existing legacy systems. The CIP component addresses the task of integration and a vast number of other sophisticated processes.

The project has implemented several system architectures from the reference architecture. One example application is the agent-based system that was developed and tested for Oslo International Airport. The system architecture is exactly the same as depicted in Example 3 in Figure 5, with Oslo airport's own content management system, WebCentral 2000, incorporated on the CSP-side. Some screenshots of the

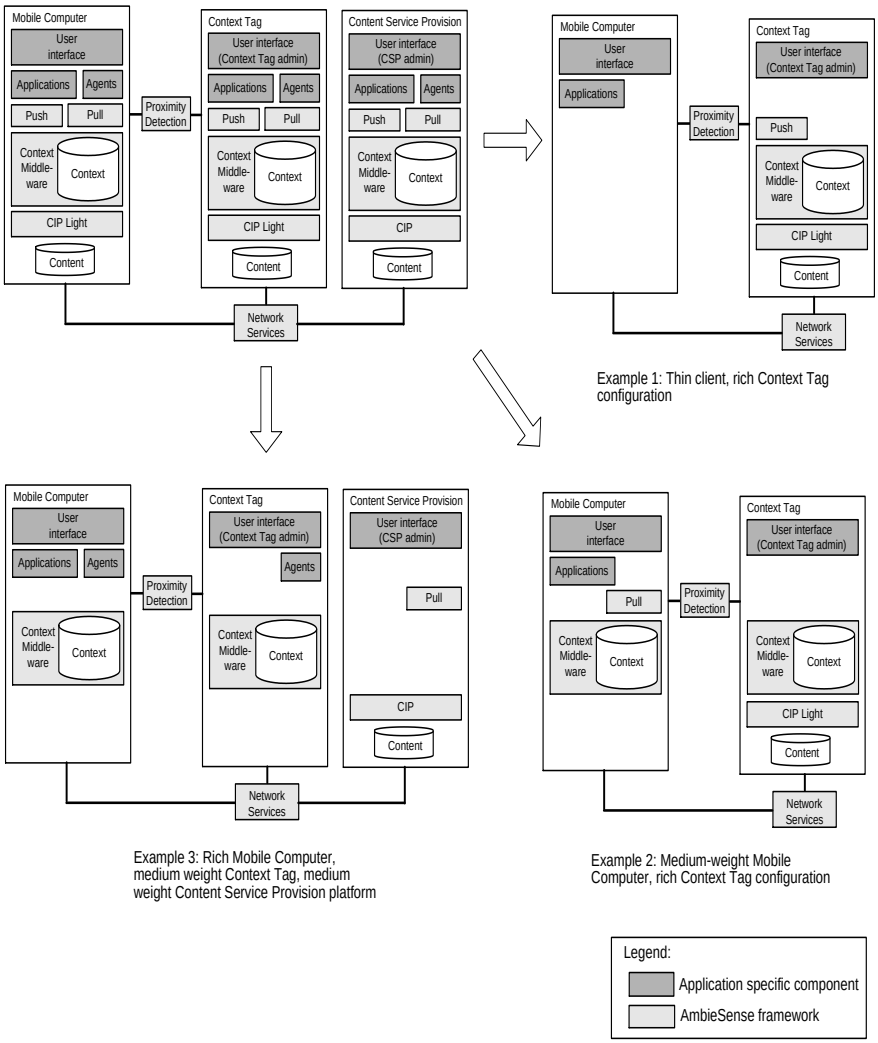

Figure 5: Example system architectures instantiated from reference architecture

Oslo airport application is found in Figure 6.

Even with such a rich client on the PDA, the client application responds with immediate recommendations to the user once the user context (of which preferences are one part) was changed. The agents provide personal recommendations to the user based upon the user contexts (i.e. preferences, flight, and location within airport). The content offered is the same as all travellers get at the airport, including special offers, shopping, flights, and transportation. The user is notified by changes in recommendations by a blinking tip button. This happens either when the user changes the personal preferences (i.e. user context), or when the user is nearby a context tag (i.e. user context is enriched by the context tag)

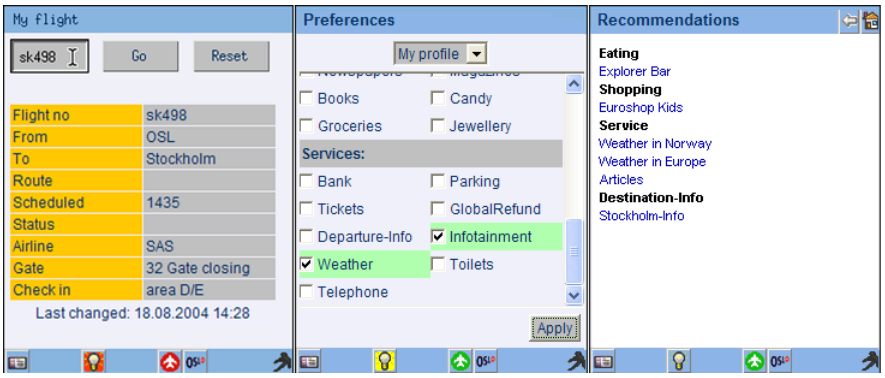

Figure 6: The Oslo airport application based on agents. A) My flight info via WLAN, B) User preferences/ context, C) Agent recommendations based on the current user context.

3 Applications and Agents

Applications in the AmbieSense Reference Architecture are developed to implement the business logic required to serve the needs of a technical solution. This application logic acts as a mediator between communication equipment, storages, middleware components and the user interface components.

Applications and agents in AmbieSense can be compared with the Controller in the Model-View-Controller paradigm. Hence, they embed the application logic of the system. They can reside on the context tags, the mobile computers, or as mediators towards the CIP located in the network. Agents can be said to accomplish a range of tasks for the user. They can be defined as self-contained, autonomous pieces of software. An AmbieSense application can be implemented using agents.

3.1 The AmbieSense Multi-Agent System and Agent Types

Within AmbieSense, part of the motivation for using agent systems is based upon non-functional requirements and lies in the fact that the complexity of the interaction of the system's users with their environment, including the sheer diversity of usage domains and contexts, requires a modular, component-based approach. The requirement for posing context sensitive requests to a distributed system implied the use of a multi-agent system architecture. Additionally, the non-functional requirements specified for the AmbieSense system, including the requirement to scalability and extendibility into new domains with new users and new contexts, made obvious the need for a MAS-like solution.

The AmbieSense Multi-Agent System (A-MAS) uses JADE/LEAP[3], an existing multi-agent framework, for the protocol and communication. The agents implemented within the JADE/LEAP framework provide the generic core functionality for handling user contexts and triggering content queries. The JADE/LEAP platform was chosen as a result of a survey of state-of-the-art systems and an evaluation process of agent programming. On top of the JADE/LEAP framework, the A-MAS integrate with intelligent components for context-based content retrieval. A method that uses Case-based reasoning for context recognition, as well as a semantic web approach to content classification enables the retrieval of relevant content to the situation.

The AmbieSense Multi-Agent System combines agent, context and content technology together. There are four agent categories, which are derived from the JADE/LEAP framework and thus benefit from that infrastructure[4]:

- *Content Agents* provide low-level content dependent functionality by interfacing directly with CIP and the underlying content providers.
- *Context Agents* are the principal agents of the A-MAS. They interact with the context middleware and administer the access to the user's context space while ensuring privacy and security. The context agent updates and maintains the user context and triggers the queries for content conducted by the content agents. The context agent will forward this context to content agents and integration agents that will convert the context to a proprietary query fit for the respective IR and content modules. The context agent keeps track of the available search types/modules and will always forward the context to each of these modules.
- *Recommender Agents* use contextual information and employ reasoning techniques for an analysis of the users' situation in order to provide appropriate content.
- *Integration Agents* are a kind of wrapper that provides interaction capabilities between the A-MAS and external (non-AmbieSense) components.

The four A-MAS agent categories in turn use services from other components, such as the push / pull, CIP, and context middleware. Additionally, agent-internal knowledge representation such as ontologies and contexts may be used by the

[3] [Bellifemine 2000], http://jade.cselt.it

[4] JADE/LEAP is compliant to FIPA, Foundation for Intelligent Physical Agents (http://www.fipa.org)

recommender agent to enhance the relevance of retrieved content. The engineering viewpoint illustrates how these components may be deployed to the different platforms. The A-MAS agents, however, reside only on the mobile computer or on the context tag in order to ensure quick and secure processing of the user's context.

4. Related work

Related work can be found in many areas, but we will only focus on work related to the area of ambient and context-aware computing, because this is the main motivation for our work.

Recently there has been much discussion about the meaning and definition of context and context-awareness. These are exemplified strongly in three recent workshops: DARPA [3], UM2001 [4], and SIGIR [5]. Context information may in general be exploited by any information service in order to improve it. Three important aspects of context can for instance be where you are, whom you are with, and what resources are nearby you. This information will often change for the mobile user.

According to the definition given within [6] "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". The concept of context is not yet well understood or defined, and there exists no commonly accepted system that supports the acquisition, manipulation and exploitation of context information.

The importance of context has also more recently been discussed for information retrieval systems. Contextual information provides an important basis for identifying and understanding users' information needs. Cool and Spink in a special issue on Context in Information Retrieval [7] provide an overview of the different levels in which context for information retrieval interest exists. Within the information retrieval field, related previous work [8] argued that a user's information needs all happen within a particular context and that context information can in general be used to improve information systems for users.

Related work can also be found in the fields of ubiquitous and context-aware computing [9]. Dey et al, [10] in a special issue on Situated Interaction and Context-aware computing provide an overview. The focus from this perspective, however, has tended to be on location-based approaches and device contexts. Examples of these can also be found in few applications for tourists.

Currently, no standard method for developing context-aware systems exists. The approach taken in AmbieSense is to unify research and work on user modelling with that of context-aware applications. We believe this approach is fruitful in order to create context-sensitive information systems for a large set of diverse user groups in the future. Most other approaches have used context either as means to adapt software, devices, and network communication, or to analyse linguistic aspects of human input to the information system.

5. Conclusions

The use of user context in ambient computing is needed for several reasons: users are increasingly mobile and require ambient computing with context-aware applications; and they need personalised information services to help them in their tasks and needs.

The challenge which ambient computing applications will face is complex. It cannot be solved easily with the current isolated approaches to wireless technology, miniaturised devices, context-aware applications, information retrieval, or user modelling. Instead, an integrated approach is needed where user focus is combined with effective information management in order to achieve ambient intelligence.

The AmbieSense project has specified a reference architecture for ambient, context-aware information environments. It has been implemented in several system architectures – one of these was briefly presented in this paper. Through these applications, we have tried out variations of the architecture. The clients on the mobile devices have varied from thin clients to rich clients. The context tags have been used with Bluetooth, WLAN, and GPRS. The software and content deployed on the tags has also varied from system to system. This is also the case for the CSP.

User tests have been conducted in both indoor and outdoor environments. The most recent test involved 75 test users at Oslo Airport during summer 2004. In general, context-aware information delivery is well accepted by the test users. Further work and analysis is being carried out on the test results.

References

1 Draft Recommendation ITU-T X.901 / ISO 10746-1: Basic Reference Model of Open Distributed Processing - Part-1: Overview.

2 Myrhaug, H. I., & Goker, A. AmbieSense - interactive information channels in the surroundings of the mobile user. Paper presented at the 10th International Conference on Human-Computer Interaction (HCI International 2003), Crete, Greece.

3 DARPA Workshop on Meaning of Context, 2001.

4 User Modeling Conference, Sonthofen, Germany. Workshop on User Modelling for Context-Aware Applications, 2001. http://orgwis.gmd.de/gross/um2001ws/papers.

5 ACM SIGIR 2004 Conference, Sheffield, UK. Workshop on Information retrieval in context, 2004. http://ir.dcs.gla.ac.uk/context/

6 Anind Dey, Gregory Abowd: Towards a Better Understanding of Context and Context-Awareness. In: Proceedings of the Computer-Human Interaction 2000 (CHI 2000), Workshop on The What, Who, Where, When, and How of Context-Awareness, The Hague, Netherlands, April 2000.

7 Cool, C., and Spink:, A. Special issue on context in information retrieval. Information Processing and Management 5, 38 (2002).

8 A. Goker. Context Learning in Okapi. Journal of Documentation, 53(1):80-83, 1997.

9 Korkea-aho, M. Context-Aware Applications Survey, Internetworking Seminar (Tik-110.551), Spring 2000, Helsinki University of Technology.

10 Dey, A., Kortuem, G., Morse, D., and Schmidt, A., Special Issue on Situated Interaction and Context-Aware Computing, Journal of Personal and Ubiquitous Computing, 5, 1, (2001).