
Context: Representation and Reasoning

Representing and Reasoning about Context in a Mobile Environment

Anders Kofod-Petersen* — Marius Mikalsen**

* Norwegian University of Science and Technology,
7491 Trondheim, Norway
anderpe@idi.ntnu.no

** SINTEF Information and Communication Technology,
7465 Trondheim, Norway
mariusm@sintef.no

ABSTRACT. Today the computer is changing from a big, grey, and noisy thing on our desk to a small, portable, and ever-networked item most of us are carrying around. This new found mobility imposes a shift in how we view computers and the way we work with them. When interaction can occur anywhere at any time it is imperative that the system adapts to the user in whatever situation the user is in. To facilitate this adaptivity we propose a two tier architecture. A middleware layer implementing a general mechanism for aggregating and maintaining contextual information. The second part offers automatic situation assessment through Case-Based Reasoning. We demonstrate a multi-agent system for supplying context-sensitive services in a mobile environment.

RÉSUMÉ. De nos jours, l'ordinateur est en passe de changer de l'objet gros, gris et bruyant sur notre bureau à un objet petit, transportable et connecté que la plupart d'entre nous transporte. Cette nouvelle mobilité impose sur notre vision des ordinateurs et la manière dont nous travaillons avec eux. Lorsque nous pouvons interagir avec les autres n'importe où, n'importe quand, il est impératif que ce soit le système qui s'adapte à l'utilisateur et à la situation, et non l'inverse. Pour faciliter cette adaptativité, nous proposons une architecture deux-tiers avec une couche logiciel personnalisé implémentant un mécanisme général d'agrégation et de maintenance d'informations contextuelles. Dans un deuxième temps, nous proposons une évaluation automatique d'une situation à partir d'un raisonnement à base de cas. Nous montrons finalement comment un système multi-agent peut fournir des services sensibles au contexte dans un environnement mobile.

KEYWORDS: Context-Aware, Middleware, Case-Based Reasoning, Multi-Agent

MOTS-CLÉS: Attention au contexte, Logiciel personnalisé, Raisonnement à base de Cas, Système multi-agent

1. Introduction

The users of mobile computers today are bringing an ever increasing amount of computational power and storage along. Most mobile users today are also equipped to access the Internet via a wide array of different carriers. With this movement of the computer from the desktop to the ubiquitous paradigm, as described by Weiser (Weiser, 1993), the computer system should now adapt to the user's situation, instead of the user adapting to the computer.

Situation adaption, or services and products customisation/personalisation, consists of two major components. Some type of *contextual information* is required, such as information about the user, the user's environment, etc. However this is not sufficient; some mechanism to *reason* about this information is also required (reasoning ranges from the simplest stimuli-respond systems, to complex deliberating systems). When these two components are present, a *context-aware* system will be able to deliver *context-sensitive* services to the users.

Even though a lot of research has been conducted within context-aware systems, the core term *context* is not yet a well defined concept; in particular what the term *context* really describes. More often than not, context is defined *ad hoc* from project to project. As a consequence of this the interesting issue of knowledge use in context-aware systems is almost invisible. Most of the research until now have been focused on the technical issues associated with context, and the syntactic relationships between different contextual concepts.

This article describes an approach to automatic situation assessment in a mobile environment within the AmbieSense research project¹. It describes a two tier system. One part is the Context Middleware where simple context-aware applications can be realised. The second part utilises Case-Based Reasoning to implement more complex situation assessment. Even though the AmbieSense system includes both mobile and fixed parts, this article focuses on the mobile part of the system. The three major issues covered here are: the open-ended context model, the multi-agent system, and the reasoning mechanism.

This article is arranged as follows: First, recent work, related to the above problems will be presented. Secondly, the AmbieSense system will be described, including the architecture, context model, and reasoning mechanism. Section 4 will demonstrate an example of usage. This is followed by a short presentation of a user-acceptance test executed at the main airport in Norway. Finally, the conclusion and outlook on future work will be described.

1. <http://www.ambiesense.com>

2. Related Work

Definition of context used in the literature falls into two main categories: those who define context by specific entities, such as location or object; and those who look at context from a more conceptual viewpoint, and focuses on the relationships and structure of contextual information (the formal approach).

The term *context-aware* was first used by (Schilit and Theimer, 1994), the definition by Schilit and Theimer falls into the first category. They looked at location as the primary part of context. However, a notion of the identity of nearby people and objects were also included. This notion of context being more than just location was more thoroughly described in (Schilit et al., 1994).

The research on context seen from examples includes lots of work, such as: Brown et al. (Brown et al., 1997) who defined context as location, identity of nearby people, time of day, etc; Ryan et al. (Ryan et al., 1998) reported on a fieldwork where they viewed context as location, environment, identity, and time. The list of projects using context by enumerating examples are very long, yet an overview of some projects can be found in (Chen and Kotz, 2000).

Beside this operational view on context, some researchers attempts to more formally define context. Schmidt et al. (Schmidt et al., 1999) defined context as: "knowledge about the user's and IT device's state". The well known definition from Dey (Dey, 2000) stated that: "Context is any information that can be used to characterize the situation of an entity". Both of these views focus on the idea that context is some particular type of information. Brézillon and Pomerol take the view that there is no special type of knowledge that can objectively be called context, they argue that context is in the eye of the beholder, or in their own words: "... knowledge that can be qualified as "contextual" depends on the context!" (Brézillon and Pomerol, 1999)

As stated above, the research field of context-awareness has seen a lot of research covering many diverse topics. A large portion of this research is relevant to the research presented here. However, we have decided to focus on two projects relevant to our research: the *Context Toolkit* by Dey (Dey, 2000), and the *Reflective Middleware Solution* by Capra et al. (Capra et al., 2001). These two project has the closest resemblance to our middleware solution for acquisition and aggregation of context:

The Context Toolkit (Dey, 2000) aims to add the use of context to existing non-context-aware applications and to evolve existing context-aware applications.

The Context Toolkit introduce the context widget, which is responsible for acquiring a certain type of context information, and make this information available to applications. Applications access the widget by using poll and subscribe methods. Context widgets operate independently from the applications that use them. Context widgets make the distribution of context sensing devices in the architecture invisible to the context-aware applications, mediating all communication between applications and components.

The context interpreter incorporates interpretation functionality to try to predict future actions or intentions of users. The interpreter accepts one or more contexts and produces a single piece of context. One example may be to get all contexts from all widgets in a conference room, and determine that a meeting is occurring. This functionality requires the programmer to write the actual code that performs the interpretation for this specific problem.

The Context Toolkit delivers a standardised way of implementing the syntactical part of context-aware systems. However, the reasoning about contextual information must be implemented for each domain and application. Thus, making it flexible only in the design and implementation phase, and not in run-time.

Another approach to aggregate context has been developed by by Capra et al. (Capra et al., 2001). In this system a marriage of reflection and metadata is suggested as a mean to create a middleware that give applications dynamic access to information about their execution context.

In this view, middleware is seen as a network middleware. Network middlewares sit on top of a network operating system and provide application developers with higher levels of abstractions, hiding complexities introduced e.g., by distribution (e.g., disconnections).

Network middlewares have been designed and work successfully on stationary computers, but they do not appear to be suitable for the mobile setting for the following reasons: Firstly, the interaction primitives (e.g., distributed transactions) assume high-bandwidth connection of components, as well as constant availability. This is not the case in mobile systems, where unreachability and low bandwidth is the norm rather than an exception. Secondly, completely hiding implementation detail from applications makes little sense. Mobile systems need to detect and adapt to drastic change occurring in the environment, such as battery power. If we have complete transparency, the middleware need to make decisions on behalf of the application. Applications however, make more efficient and better quality decisions based on application specific information.

Reflection and meta data are used to build the system that supports context aware applications. Applications pass metadata to the middleware. This metadata constitutes a policy as to how the applications want the middleware to behave as a result of a specific context occurrence. As context and application needs changes continuously, one cannot assume that meta data are static, therefore applications use reflection mechanisms offered by the middleware to inspect their own meta data, and possibly alter it according to changing needs. The meta data is standardised using XML Schemas.

3. AmbieSense

AmbieSense is a project in the Information Society Technologies (IST) Programme of the European Union. The goal is to develop a set of software and hardware tools to

facilitate context aware computing. The AmbieSense framework can be used to build ubiquitous information channels in the surroundings, capable of delivering the right information at the right time to the mobile user.

3.1. System overview

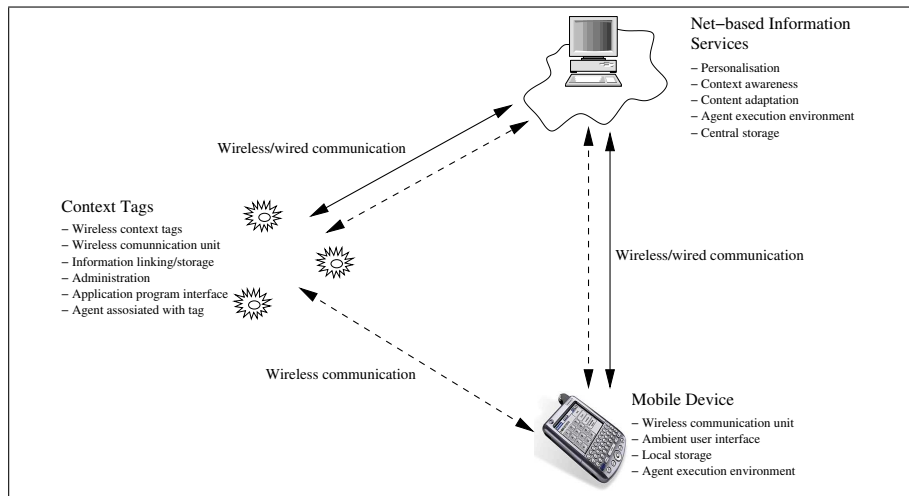


Figure 1. Overall Architecture

Figure 1 suggests how AmbieSense implements ubiquitous information channels. The AmbieSense framework consists of three major hardware components: the mobile device (PDAs and smart phones), the Context Tag (small, Bluetooth enabled computers), and net-based information services.

Information is available from the net-based information services. Information on these services is typically published by some Information Provider. Information Providers using a system such as AmbieSense want to increase the value of their information by personalising the information offered to the mobile user.

The Context Tag, a small Bluetooth enabled computer, can supply information (contextual and other) to a mobile user from the information providers. Contextual information can, as an example, be location. It can also distribute localised information, such as the menu from the particular restaurant where it is mounted.

The mobile system is divided into three major parts: the Context Middleware (Section 3.3), the multi-agent system (Section 3.4), and the reasoning mechanism (Section 3.5).

3.2. Context

Given the domain independent nature of the AmbieSense system, a broad and non-excluding definition of context is needed. We extend the definition of context given by Dey (Dey, 2001), applying the following definition:

Context is the set of suitable environmental states and settings concerning a user, which are relevant for a situation sensitive application in the process of adapting the services and information offered to the user.

At the same time we agree with the view advocated by Brézillon and Pomerol (Brézillon and Pomerol, 1999); that context is not a special kind of knowledge. Hence, particular kinds of knowledge can be considered context in one setting and domain knowledge in another. We believe that this pragmatic definition of context allows application developers to efficiently rule out information that is not context in their particular application domain (or their context). At design time, developers can ask the question; is this information relevant for adapting our services and information? If the answer is no, the information is discarded as not being context, and excluded from the context model in the Context Middleware. This flexibility leads to an open context model that only defines the taxonomic structure in the design phase (see Figure 2).

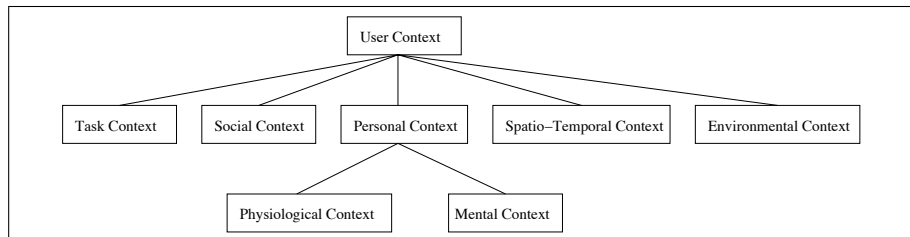


Figure 2. *User Context in the AmbieSense project*

Even though we also argue a knowledge model, where context is not a special type of information, structuring our knowledge model is still essential. Since we are focusing on applications utilising contextual information to improve services provided to users, we have chosen to structure our model around a taxonomy inherited from the context-aware tradition.

The context is divided into five sub-categories (a more thorough discussion can be found in (Göker and Myrhaug, 2002)): *i*) Environmental context: This part captures the users surroundings, such as things, services, light, people, and information accessed by the user. *ii*) Personal context: This part describes the mental and physical information about the user, such as mood, expertise, disabilities and weight. *iii*) Social context: This describes the social aspects of the user, such as information about friends, relatives and colleagues. *iv*) Task context: the task context describe what the user is doing, it can describe the user's goals, tasks, activities, etc. *v*) Spatio-temporal

context: This type of context is concerned with attributes like: time, location and movement. The different aspects of the contexts are attribute-value tuples that are associated with the appropriate contexts.

The model depicted in Figure 2 shows the top-level ontology. To enable the reasoning in the system this top-level structure is integrated with a more general domain ontology, which describes concepts of the domain (e.g., Airport Hall, Gate, Restaurant, Newsstand) as well as more generic concepts (Task, Goal, Action, Physical Object) in a multi-relational semantic network. The model enables the system to infer relationships between concepts by constructing context-dependent paths between them. One important use of this is to be able to match two case features that are syntactically different, by explaining why they are similar ((Aamodt, 1994), (Jære et al., 2002)). For example, the concept Magazine matches Journal in the context of Newsstand.

A part of the domain model—in which the context model is integrated—is illustrated in Figure 3.

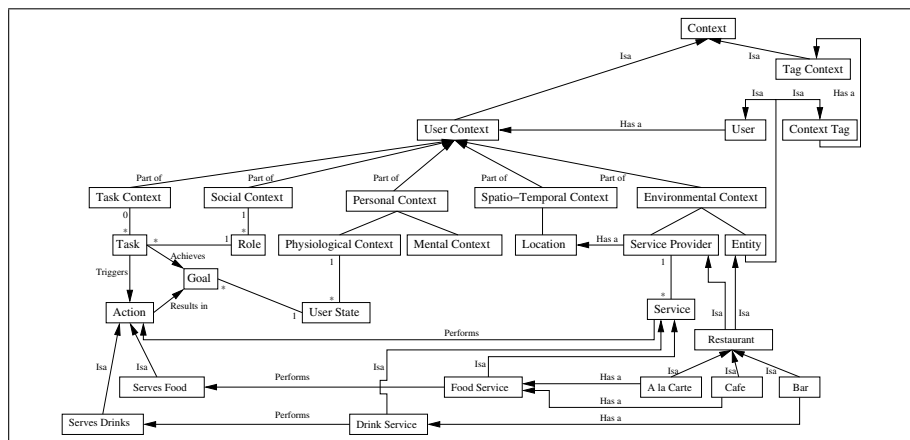


Figure 3. *Enriched Context Model*

This work postulates that there exists a goal or task in any situation. It would be futile to identify a situation unless there is some task connected to it - no matter how mundane. This is most obvious when dealing with users, where a situation implies that there is a problem that needs to be solved; such as the possible situation "hungry user", which implies the goal of *not hungry user*, leading to the task *provide food*, with a subtask *locate food*.

3.3. Context Middleware

The utility of context aware services has already been demonstrated by (Bristow et al., 2002). The problem today is that many context aware applications custom build proprietary infrastructures for their context management (Lei et al., 2002).

The main contribution of the context middleware is to provide a generic and user-friendly context management infrastructure, which collects and maintains context. The context middleware allows applications to utilise relevant context information, paying little or no attention to the details of context management.

The context information available in the context middleware is provided from some context sources. Examples of sources are mobile users (through a user interface), applications, software agents and sensors (such as the context tag). Clients may be other context sensitive applications or other software agents. The sources and clients interact with the context middleware, using services relevant to their particular needs, without integrating with each other directly.

There are essentially three categories of features that a context aware application can support (Dey, 2001); i) *presentation* of information and services to a user; ii) automatic *execution* of a service for a user; and; iii) *tagging* of context to information to support later retrieval. The following sub-sections summarise the core services that the context middleware offers to context aware applications.

3.3.1. Storing and retrieving Context in Context Spaces

The context middleware implements a context space (Myrhaug, 2001). The context space abstraction is essential to capture the difference between transient and persistent context (Lei et al., 2002).

Transient context reflects the environment at a single point in time, whereas persistent context represents a recurrent pattern of transient context. The fact that Hans Inge eats a Danish pastry at 10:30 AM in the morning is transient context, and the fact that he eats a Danish every morning at 10:30 is persistent context. A context space consists of a context history, a current context and a context future, populated by context instances.

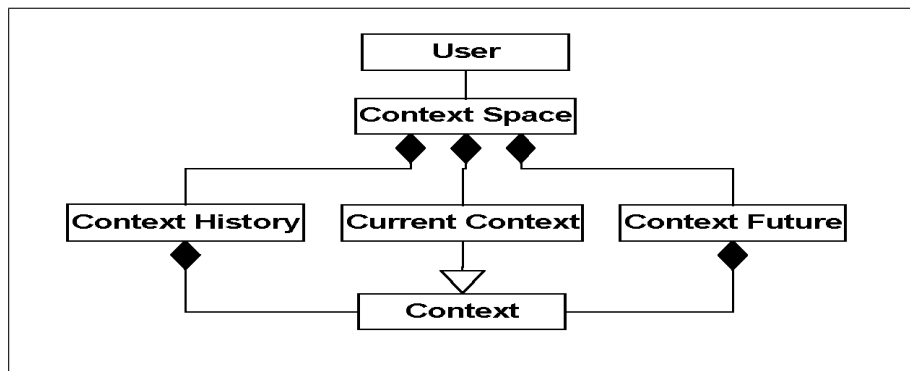


Figure 4. Architecture of a Context Space

Context history can help applications to predict actions and intentions of users based on their previous (stored) contexts. Once deduced, this can be put in the context

future (context cache). The current context is the context with corresponding attribute-value tuples currently relevant. When no longer relevant, it can be stored in the context history for future reference. From the context history, clients can retrieve contexts using the context middleware search routines.

3.3.2. *Avoiding Infinite Definitions of Context; generating valid Context Representations using Context Templates*

Given the generic context definition, the taxonomic nature of the context model, and the domain independent nature of the context middleware, there is a risk of infinite definitions of context; a situation where everything can be context. To remedy this, we suggest the use of context templates to create context patterns that constrain context representations and define contextual information in a domain dependent way. This has the same rationale as any other domain model. Using the context template, application developers define what context structure (contexts with sub contexts), permissible attributes and valid values a context can have. Context instantiations that comply with the pattern laid out by the context template are valid context instances. Consequently, the context template defines what is context, and efficiently avoids infinite definitions of context.

By using the built in context validation tool of the context middleware, one has the measures to ensure that a given context instance is valid toward a context template.

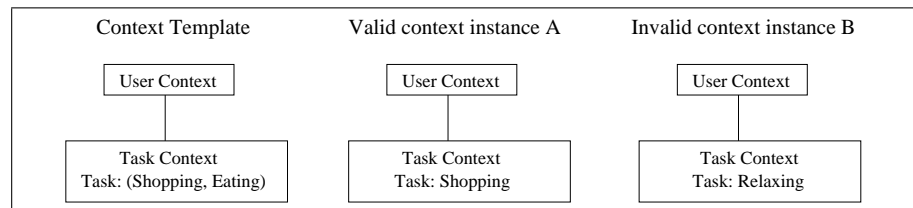


Figure 5. *Example context template and context instances (valid (A) and invalid (B))*

3.3.3. *Context publishing and subscription*

In order to properly adapt information and services, it is important to the context aware clients to be continuously notified of changes that occur in context. To accomplish this, clients use subscription mechanisms of the context middleware to indicate that they are interested in notification when context changes as a result of context sources publishing new context into the current context of a context space. Clients can subscribe to several kinds of changes: i) changes of the entire current context, i.e., it is replaced by an entirely new current context; ii) changes in a sub-context in the overall context structure, and; iii) changes in the attribute-value level in the context representation (e.g., be notified when location changes in the spatio-temporal context).

3.3.4. *Tagging Context to Information*

As mentioned above, tagging of context to information for later retrieval is a core task for context aware applications. This is explicitly supported by the context middleware, allowing application developers to create links from contexts, attributes and values, to relevant information. As a context is entered into the context space (e.g., context history), it can be linked to a relevant information entity. When this context is retrieved from history for some reason, the information entity is likely to be of relevance again, and can be reused.

3.3.5. *Merging Context from various Context Sources*

As context flows into a context space from various sources, applications need a mechanism to integrate this information into one common context representation. The context merging service allows applications to define what parts of a context that are subject to change (e.g., location in the spatio-temporal context), and can be merged with other context representations. Using the validation mechanism and a context template, one can assure that two context representations have the same structure and attribute-value tuples, and consequently can be merged correctly.

3.4. *Agents*

As a part on the AmbieSense system, the mobile device holds the agency that identifies user situations and solves the problems associated with the current situation.

Agents have been chosen for various reasons. One of the most important is the modularity that an agency supports. This flexibility is a great benefit when new application agents are implemented into the system. The flexibility lays primarily in the idea that the implementation of the agent's behaviour is hidden for other parties. This eases the burden when implementing additional agents, since the programmer only needs to relate to the behaviours of other agents, conventions, and language spoken in the agency.

This project utilises the Jade (Bellifemine et al., 1999) platform. This platform was chosen by the AmbieSense project for various reasons. One reason that is worth mentioning in this context is the fact that the Jade platform is FIPA compliant. When an agency is FIPA compliant it adheres to the standards for heterogeneous agent interacting. One particular benefit of using a FIPA compliant agent implementation, is the use of a standardised Agent Communication Language (ACL), which further eases the programming burden when making additions to a system.

The agency consists of three kinds of agents:

- The framework agents supplied by the agent platform Jade
- The core agents handling the situation detection, and goal decomposition, which in turn are connected to the context middleware via the Context Agent.

- The application agents, each handling one specific application, such as map construction or news gathering.

The Context Agent communicates with the Context Middleware, which maintains the space of contextual data. This agent receives new *current contexts* from the Context Middleware, translates them to Jade ACL messages, and sends them to the relevant receivers. This translation is achieved through the use of the ontology described in Section 3.2. The ontology is available to all agents running in the agency

The primary relevant receiver in this work is the Case-Based Reasoning agent that utilises the Creek (Aamodt, 1991) Case-Based Reasoning system to identify the current situation (see Section 3.5). This agent also incorporates the ontology for reasoning about concepts. As an example, *Isa* relations will be followed to reason about unknown concepts.

Based on this identification, the Case-Based Reasoning agent notifies the Task Facilitator about this particular situation. This notification is a set consisting of the situation with the corresponding contextual information, and the goal associated with it. Once the Task Facilitator has reached it's goal, the solution will be returned to the context agent.

The Task Facilitator uses the Unified Problem-solving Method description Language (UPML) (Fensel et al., 2003) to maintain an overview of the application agents' services and for handling the task decomposition. It will receive the situation and task description from the Case-Based Reasoning agent, decompose the task into the different sub-tasks required, and recruit the correct application agents for solving the tasks.

Application agents are responsible for solving their own particular tasks. At present four different application agents exists:

- Map agent, who can access the map server and supply map suited to the particular context information.
- News agent, who can gather relevant news.
- Information agent, who can gather information generally available on the net.
- Airport agent, who can solve airport-related problems.

Applications agents are the type of agents where it is most likely that more are to be introduced. When a new application agent is to be introduced into the system, the programmer needs only to know the context ontology, and how to specify the capabilities of this agent in UPML, and the agent is basically ready to be a part of the system.

3.5. Reasoning

In most of the research in context-aware systems, the problem of filtering the vast amount of contextual information that is available, in such a way that the identifica-

tion of important constellations of the contextual information is feasible, has not been thoroughly addressed. Case-Based Reasoning is a promising method for this.

Case-Based Reasoning (Aamodt and Plaza, 1994) is concerned with adapting to new situations by remembering similar earlier experienced situations (cases). This method has, some of, its roots in the work by Roger Schank on dynamic memory (Schank, 1982). Central to Schank's work is the idea of remembering earlier *situations* (cases) and using situation patterns in reasoning.

Case-Based Reasoning is a particular interesting and promising method for addressing context-awareness related to particular situations and the assessment of these situations (Zimmermann, 2003), (Kwon and Sadeh, 2004), (Plaza and Arcos, 2001). The particular version of Case-Based Reasoning employed and extended through this work is Creek (Aamodt, 1991) (Aamodt, 2004).

For this particular work we have opted for a two tier reasoning mechanism: The on-line part that resides on the user's mobile device, and the off-line reasoning that resides on the user's backbone system.

3.5.1. *On-line reasoning*

Different types of contextual information can arrive in a very diffuse fashion, e.g., time is continually flowing into the system, whereas location might be pseudo static. Since Case-Based Reasoning works on discreet cases, the continuous values flowing into the system must be made discreet. The context agent receives a *Current Context* from the Context Middleware, translates it to fit the ontology used in Jade, and sends it to the Case-Based Reasoning agent.

Once a new context arrives, the Case-Based Reasoning cycle is activated (see Figure 6). The system will try to retrieve a known context or case, and classify the current situation based on the retrieved one. When the situation has been classified, the associated goal is presented to the task decomposition agent. After the agency has satisfied the goal associated with this particular situation, the case will be stored in the case base in a triplet consisting of: *i*) the contextual information describing the situation, *ii*) the problem associated with the situation, and *iii*) the solution constructed by the application agents.

Since the user is expected to experience at least a few different situations a day, the storage of the cases will quickly fill up the mobile device. This potential vast amount of cases will also severely hamper the searching process for the Case-Based Reasoning mechanism. To remedy this, some of the reasoning process is moved into the user's backbone servers.

3.5.2. *Off-line reasoning*

There are potentially two main problems with the use of Case-Based Reasoning for identifying situations: the storage problem, and the problem of indexing and searching.

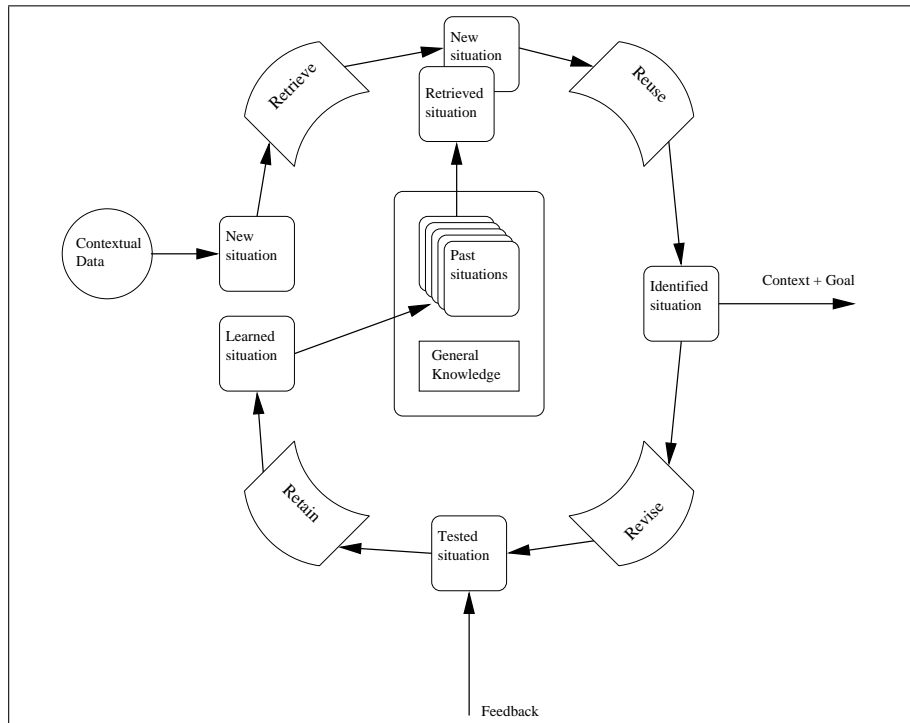


Figure 6. Case-Based Reasoning Cycle (adapted from Aamodt and Plaza (1994))

First and foremost is the problem of storing the, potentially vast, amount of cases constructed during run time. To solve this, the user has a personal persistence storage available on the user's home network. This storage is used for storing the cases, and will be synchronised when the user has an up-link. This large amount of data does not only affect the amount of storage space needed, it will also severely affect the indexing and matching algorithm used.

To remedy this a generalisation process occurs on the home net. Similar cases is grouped into prototypical cases, e.g., everything that the 600 business meetings has in common will constitute the prototypical business meeting situation. These prototypical cases will be part of the on-line case base, and be used in the every day reasoning process. Generalisation of cases is a well known research area within Case-Based Reasoning, for an overview see for an example (Maximini et al., 2003). This case base is structured by prototypes, which are generated on the basis of the amount of similar parameters in the point-cases.

4. A Hungry User in an Airport – An Example

In this scenario we will show how the system can detect that a user is hungry in OSL Airport Gardermoen and assist him in finding food to his liking.

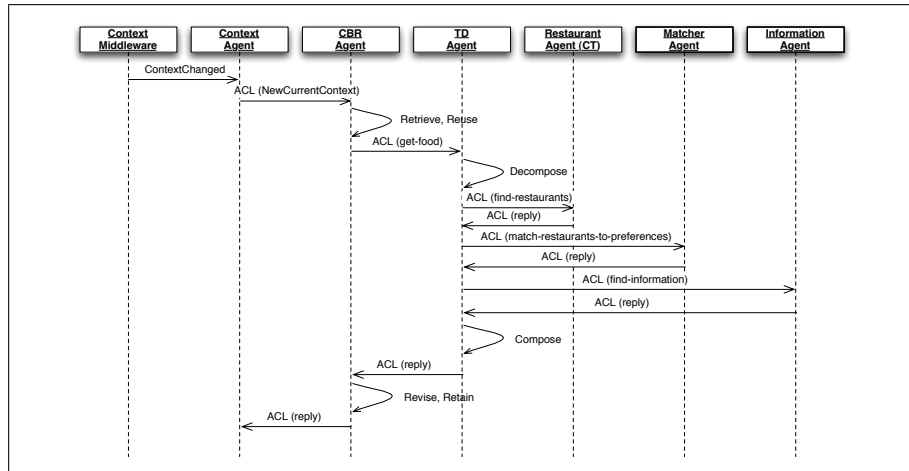


Figure 7. Sequence diagram for Hungry User example

Carrying the AmbieSense enabled mobile device, the user enters the restaurant area of OSL Gardermoen, where a context tag is mounted. From the current user context (see Figure 8) on the client, we know that the user prefers Italian food. The time attribute in spatio-temporal context shows that the time is a quarter past one.

As the user enters the context tag's Bluetooth zone, the context information on the tag is transferred to the client. This information is merged with the current user context, and represents a new context. New information in the current user context is now that the location in Spatio-temporal context is *Eating area*.

The context agent receives a new current context from the context middleware (ContextChanged). The context agent translates the context instance to a Jade ontology instance, which is sent to all interested parties, in this case the Case-Based Reasoning agent (NewCurrentContext).

The Case-Based Reasoning system can now take this situation case and try to find a matching case in its case base (Retrieve). A hungry user case is matched against existing cases, based on the information presented above. The matching case is a case that identifies a task get-food (Reuse), and can leave it to the Task Decomposer agent to accomplish it (get-food).

The Task Decomposer agent knows what agents are available on the mobile system, and what services they offer. The Task Decomposer agent can furthermore look up the services that are available in the new current context, under Service Providers in the Environment Context (i.e., available on the Context Tag). This in-

formation is then used to decompose the task of `get-food` into `find-restaurants`, `match-restaurants-to-preferences`, and `find-information`. Each task corresponds to an action performed by an agent on either the mobile device or on the Context Tag. In this scenario the agents who can handle `match-restaurants-to-preferences` and `find-information` resides on the mobile device, and the service offering `find-restaurants` resides on the Context Tag. Each of these agents or services is then send a request for information. When the results are returned the Task Decomposer Agent constructs a coherent result (`Compose`) and send a reply to the Case-Based Reasoning Agent (`Reply`).

When the Case-Based Reasoning agent receives the reply to its original request for the solution to `get-food`, it will execute the storing of this new situation (`Retain`). The methods for revising the solution proposed to the user has, as of today, not been implemented. This is due to our aim of making this solution as non-intrusive as possible. For possible solutions see Section 6. Once the new situation is stored the Case-Based Reasoning agent will send the suggested solution to the Context Agent for further user (`reply`).

Once the solution has been returned to the Context Agent, it can remind the user that it's time for lunch, by telling him that: "there is a nice Italian restaurant three minutes away. The restaurants web-page claims that the service is acceptable, and that you can get a meal for only 10 €".

Case 0 (Current Context)	Case 1 (Hungry User Situation)
TASK: identify User Need	TASK: Get Food
TASK STATE: In Process	TASK STATE: Accomplished
SOCIAL CONTEXT:	SOCIAL CONTEXT:
Role: Departing	Role: Departing, Hungry
PERSONAL CONTEXT:	PERSONAL CONTEXT:
PREFERENCE:	PREFERENCE:
Italian Food	Italian Food
Guardian Newspaper	Le Figaro Newspaper
SPATIO-TEMPORAL:	SPATIO-TEMPORAL:
Current Location: OSL – Eating Area	Current Location: Eatin Area
Current Time: 13:15:00	Current Time: range[11:00 – 14:00]
ENVIRONMENTAL:	ENVIRONMENTAL:
Entity: OSL-CT-TAG-12	
Services: find-restaurants@OSL-CT-TAG-12	

Figure 8. *Current Context and Hungry User Case*

5. Test case

During week 33 in 2004, the AmbieSense technology was tested at Oslo Airport Gardermoen (OSL Gardermoen is Norway's largest airport and a member of the AmbieSense consortium). The technology was tested on two different hardware platforms; mobile phones and PDAs. A total of 75 users participated in testing; 55 tested the mobile phone system while 20 tested the PDA application. 18 Context Tags were placed in the airport: 11 for the mobile phone system, and 7 for the PDA system. Four mobile phones and two PDAs were used. The two systems worked independently, without interference.

The test at OSL Gardermoen was conducted as a field study where testing was performed in the real-world environment, where such technology as AmbieSense presumably will be used.

In testing the AmbieSense system, Scenario-Based Testing (SBT) was applied (Preece et al., 1994). The user acceptance test focused on what the users did instead of on how the product (technology) performed. We captured representative tasks (via use cases) that the users would perform, and then applied them and their variants to the test. Using observations and interviews, we gathered considerable feedback of the users' acceptance of AmbieSense technology.

The PDA part of the test focused on personalisation at an airport. Personalisation was accomplished as an implementation of the overall architecture (see Figure 1). The implementation included targeted applications with a graphical user interface allowing users to edit their personal context and receive personalised information from the system. The applications utilised Context Middleware instantiations that were running on both mobile clients and the Context Tag. The agency consisted of information retrieval agents that retrieved relevant information as the context of the user dynamically changed. By editing their preferences, the users would get different results from the same Context Tag. Furthermore, as they moved from tag to tag, the recommendations from the agents would change according to their current context (a new location is one of the parameters that trigger a change in context).

In the interviews we found that personalisation is a key asset in making people using systems like AmbieSense. However, based on the received feedback, it seems that a correct balance with regard to privacy concerns must be found. User interests and preferences are important parts of the personalisation of information. People seem reluctant to give away personal information like name, address, email etc. Yet age, gender, hobbies and interests are examples of information most users are willing to share.

Users at an airport are generally positive to the notion of using intelligent agents to help them filter what information that should be presented. This must be seen in relation to the "kill time versus save time problem", that is; if people are in a hurry, the information that is presented to them should be personalised. However, if they have a

lot of time to spend at the airport, they might wish to use some of this time browsing through the information manually.

6. Conclusion and further work

The work presented here describes an approach to handle three important issues in context-aware applications: namely the possibility of specifying behaviour based on context in the implementation phase, the problem of aggregating contextual data from many and diverse sources, and the issue of reasoning about context in run-time. Furthermore, the context middleware handles the aggregation of contextual information from many sources and presents it in a standardised manner to the reasoning part of the system.

As stated earlier it is not sufficient to gather and aggregate contextual data, some kind of reasoning about this data is required. This work proposes the use of Case-Based Reasoning as a method for context-sensitive applications. As this is ongoing research, experimental results are still preliminary. However, work by Zimmermann (Zimmermann, 2003) suggests that Case-Based Reasoning is a promising method for identifying the correct combination of contextual information that leads to a good situation understanding.

One of the main issues to be resolved is the question of feedback to the system from the user. For Case-Based Reasoning to function properly some kind of verification/falsification of the results must occur. Today the only method is to ask the user. However, to insure the calmness of this technology other feedback mechanisms must be investigated. Presently, the possibilities of looking at time-series in case base is under exploration. It is possible to check if a user chose to behave in accordance with suggestions given by the Case-Based Reasoning mechanism by looking at situations occurring later on. For example, if the hungry user did indeed visit the Italian restaurant suggested in our example, it is possible to discover that fact by looking at situations occurring after the suggestion was made; thereby learning that this suggestion was a valid one. If, on the other hand, the user visits the Chinese restaurant it is reasonable to assume that the suggestion made was wrong.

Acknowledgements

This work is part of the AmbieSense project, which is supported by the EU commission (IST-2001-34244).

References

- Aamodt A., A knowledge-intensive, integrated approach to problem solving and sustained learning. PhD thesis, University of Trondheim, Norwegian Institute of Technology, Department of Computer Science, May 1991. University Microfilms PUB 92-08460.
- Aamodt A., “Explanation-driven case-based reasoning”, In S. Wess, K. Althoff, and M. Richter, editors, *Topics in Case-based reasoning*, pages 274–288. Springer Verlag, 1994.
- Aamodt A., “Knowledge-intensive case-based reasoning in creek”, In P. Funk and P. A. G. Calero, editors, *Advances in case-based reasoning, 7th European Conference, ECCBR 2004, Proceedings*, pages 1–15, 2004.
- Aamodt A., Plaza A., “Case-based reasoning: Foundational issues, methodological variations, and system approaches”, *AI Communications*, 7(1):39–59, 1994.
- Bellifemine F., Poggi A., Rimassa G., Jade - a fipa-compliant agent framework. Technical report, Centro Studi e Laboratori Telecomunicazioni, 1999. Part of this report has been also published in Proceedings of PAAM’99, London, April 1999, pagg.97-108.
- Brézillon P., Pomerol J.-C., “Contextual knowledge sharing and cooperation in intelligent assistant systems”, *Le Travail Humain*, 62(3):223–246, 1999.
- Bristow H. W., Baber C., Cross J., Wooley S., “Evaluating contextual information for wearable computing”, In *Proceedings of the Sixth International Symposium on Wearable Computers*. IEEE Computer Society, 2002.
- Brown P. J., Bovey J. D., Chen X., “Context-aware applications: From the laboratory to the marketplace”, *IEEE Personal Communications*, 4(5):58–64, 1997.
- Capra L., Emmerich W., Mascolo C., “Reflective middleware solutions for context-aware applications”, In A. Yonezawa and S. Matsuoka, editors, *Metalevel Architectures and Separation of Crosscutting Concerns - Proc. of Reflection 2001*, pages 126–133. Springer Verlag, 2001.
- Chen G., Kotz D., A survey of context-aware mobile computing research, Technical Report TR2000-381, Department of Computer Science, Dartmouth College, 2000. URL <ftp://ftp.cs.dartmouth.edu/TR/TR2000-381.ps.Z>.
- Dey A. K., Providing Architectural Support for Building Context-Aware Applications, PhD thesis, College of Computing, Georgia Institute of Technology, December 2000.
- Dey A. K., “Understanding and using context.”, *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

- Fensel D., Motta E., van Harmelen F., Benjamins V. R., Crubezy M., Decker S., Gaspari M., Groenboom R., Grosso W., Musen M. A., Plaza E., Schreiber G., Studer R., Wielinga R., “The unified problem-solving method development language upml”, *Knowledge and Information Systems Journal (KAIS)*, 5(1), 2003.
- Göker A., Myrhaug H. I., “User context and personalisation”, In *Workshop proceedings for the 6th European Conference on Case Based Reasoning*, 2002.
- Jære M. D., Aamodt A., Skalle P., “Representing temporal knowledge for case-based prediction”, In *Advances in case-based reasoning*, 6th European Conference, EC-CBR 2002, Lecture Notes in Artificial Intelligence, LNAI 2416, pages 174–188. Springer Verlag, September 2002.
- Kwon O. B., Sadeh N., “Applying case-based reasoning and multi-agent intelligent system to context-aware comparative shopping”, *Decision Support Systems*, 37(2): 199–213, May 2004.
- Lei H., Sow D. M., Davis J. S. I., Banavar G., Ebling M. R., “The design and applications of a context service”, *Mobile Computing and Communications Review*, 6(4): 44–55, 2002. ACM SIGMOBILE.
- Maximini K., Maximini R., Bergmann R., “An investigation of generalized cases”, In K. D. Ashley and D. G. Bridge, editors, *ICCBR 2003, Case-Based Reasoning Research and Development*, LNAI 2689, pages 261–276. Springer-Verlag, 2003.
- Myrhaug H. I., “Towards life long and personal context spaces”, In *Workshop on User Modelling for Context-Aware Applications*, 2001. Available at: <http://orgwis.gmd.de/gross/um2001ws/papers/myrhaug.pdf>.
- Plaza E., Arcos J.-L., “Context-aware personal information agents”, In *Cooperative Information Agents V*, number 2128, in Lecture Notes in Artificial Intelligence, pages 44–55. Springer Verlag, 2001.
- Preece J., Rogers Y., Sharp H., Benyon D., Holland S., Carey T., *Human-Computer Interaction*, Assison-Wesley, 1994.
- Ryan N., Pascoe J., Morse D., “Enhanced reality fieldwork: the context-aware archaeological assistant”, In V. Gaffney, M. van Leusen, and S. Exxon, editors, *Computer Applications and Quantitative Methods in Archaeology*, British Archaeological Reports. Tempus Reparatum, October 1998.
- Schank R., *Dynamic memory; a theory of reminding and learning in computers and people*, Cambridge University Press, 1982.
- Schilit B. N., Adams N., Want R., “Context-aware computing applications”, In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*, pages 85–90. IEEE Computer Society Press, December 1994.
- Schilit B. N., Theimer M. M., “Disseminating active map information to mobile hosts”, *IEEE Network*, 8(5):22–32, September/October 1994.

Schmidt A., Aidoo K. A., Takaluoma A., Tuomela U., Laerhoven K. V., de Velde W. V., “Advanced interaction in context”, In *Proceedings of First International Symposium on Handheld and Ubiquitous Computing, HUC’99*, pages 89–101. Springer Verlag, September 1999.

Weiser M., “Some computer science issues in ubiquitous computing”, *Communications of the ACM*, 36(7), 1993.

Zimmermann A. Context-awareness in user modelling: Requirements analysis for a case-based reasoning application. In K. D. Ashley and D. G. Bridge, editors, *ICCBR 2003, Case-Based Reasoning Research and Development*, LNAI 2689, pages 718–732. Springer-Verlag, 2003.

Anders Kofod-Petersen is affiliated with the Division of Intelligent Systems at the Department of Computer and Information Science at the Norwegian University of Science and Technology. He is currently pursuing his Ph. D. in computer science, where his work is focused on situation understanding in Context-Aware systems through the user of Case-Based Reasoning. Kofod-Petersen has a background from Behaviour Based Robotics and Evolutionary Systems. He is currently co-leading a small Research and Development company working with Semantic Web technologies.

Marius Mikalsen is a researcher at SINTEF Information Communication and Technology. He graduated as Master of Science in Information Technology from the Norwegian University of Technology and Science in 2002, with a thesis focusing on the use of intelligent agents in mobile user interfaces. Since then his research has focused on mobile systems, and how context can be used to increase the usability of such systems.