

# An Architecture Supporting Implementation of Context-Aware Services

Anders Kofod-Petersen<sup>1</sup> and Marius Mikalsen<sup>2</sup>

<sup>1</sup> Department of Computer and Information Science,  
Norwegian University of Science and Technology,  
7491 Trondheim, Norway  
anderpe@idi.ntnu.no

<sup>2</sup> SINTEF Information and Communication Technology,  
7465 Trondheim, Norway  
mariusm@sintef.no

**Abstract.** This work describes an approach to facilitate use of contextual information to improve the quality of service in a mobile ambient environment. The approach advocated here defines two levels of applications: a simple stimuli-response based system using the Context-Middleware, and a more complex and adaptive version that utilises a multi-agent environment to supply information and context to mobile users. The main goal of this work is to show how a developer easily can implement a simple responsive system, and by following simple specification of agent communications, the local ontology and complying with the specifications of the Context Middleware, develop adaptive context-aware applications.

## 1 Introduction

As the modern human is getting more and more mobile, carrying an ever increasing amount of computational power around and having access to online services, the need for personalised and adaptive information services is rapidly increasing. Even though the ubiquitous computing paradigm suggested by Weiser [1] is still a good way into the future we are seeing an ever increasing amount of work bringing us closer to this goal.

Even though the ubiquitous computing paradigm is by nature ubiquitous, it is to be expected that real world context-aware applications adhering to this paradigm is first to be introduced in limited and closed domains. We have already seen a number of applications within the tourist guide domain, such as [2] and [3].

To introduce context-aware applications in any domain they must be both usable and feasible to develop. This work presents an architecture and an implementation which allows developers to augment existing applications or to develop new context-aware applications.

This paper is organised as follows: First, some recent work related to our work is presented. Secondly, the context model employed here will be described. Thirdly, the system architecture, including the agency and reasoning capabilities currently implemented, will be described. This is followed by an example describing how a content provider can introduce new services using this architecture. Finally, the conclusion and some pointers for future work will be presented.

## 2 Related Work

Context-aware applications are a fast growing research area with a lot of diverse research. Due to the constraints on this paper we have chosen to focus on three different works, which are particularly related to ours. We have chosen to focus on the Context Toolkit by Dey [4], network middleware by Capra et al. [5], and the Context Broker Architecture by Chen et al. [6].

The Context Toolkit is concerned with construction of a framework for maintaining a generic interface to devices or pieces of software supplying contextual data. This toolkit contains a combination of features and abstractions to support context-aware application builders. The toolkit aims to add the use of context to existing non-context-aware applications and to evolve existing context-aware applications.

Using context widgets, context interpreters and context aggregators, the lower level context details are hidden from application developers. The context widget is responsible for acquiring a certain type of contextual information, such as noise, and make this information available to applications. Applications access the widget by using poll and subscribe methods. 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. Context aggregators collect the context about an entity (e.g. a person) from available context widgets and interpreters, and behave as proxies to context for applications.

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.

Capra et al. [5] designed a system for aggregation context. In their view, middleware is seen as a network middleware. Network middlewares sit on top of a network operating system and provides application developers with higher levels of abstractions, hiding complexities introduced e.g. by distribution (e.g. disconnections). This is conceptually close to the Context Middleware, used in our work, where a distributed network of contextual sensors are made available in a coherent and transparent way.

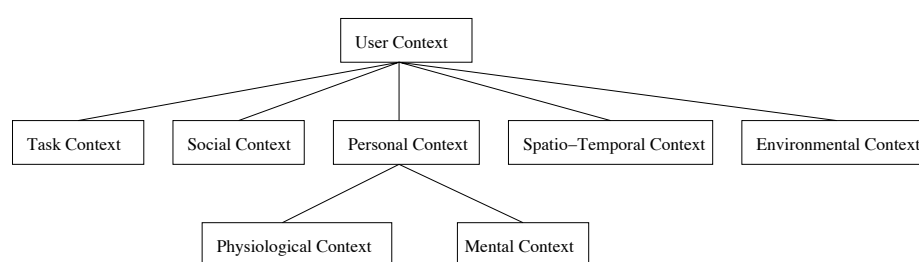
Chen et al. [6] talks about their Context Broker Architecture (CoBrA), where an intelligent broker agent maintains a shared model of the context data available. This model also assumes the responsibility for aggregating contextual data and making it available. The main difference between this model and the Context Toolkit is that the CoBrA also includes a knowledge model of the contextual information available, and in that sense facilitates more distributed reasoning capabilities for agents wishing to use the CoBra. This is in line with the knowledge perspective on context advocated in this paper.

## 3 Context

As our system has been designed to be domain independent a broad and non-excluding definition of context is paramount. We extend the definition given by [7], leading to 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.

This definition could be biased toward dividing the world into two distinct part: the context and non-context world. However, we advocate the view put forth by Brézillon and Pomerol [8], where context is not viewed as some special kind of knowledge. Rather, it depends on the situation if some particular type of knowledge is to be viewed as context or not, in their own words: "... it appears that knowledge that can be qualified as "contextual" depends on the context!" (ibid, p. 7). In other words, context can be seen as a set of invariants of a situation, used to focus the selection of variants.



**Fig. 1.** Context Structure

Even though we argue that context is not to be viewed as any particular type of knowledge, it is important that some structure is imposed. Since the focus in this work is on enhancing an content provider's ability to develop and use context-aware applications, we have structured our model around a taxonomy inspired by the context-aware tradition.

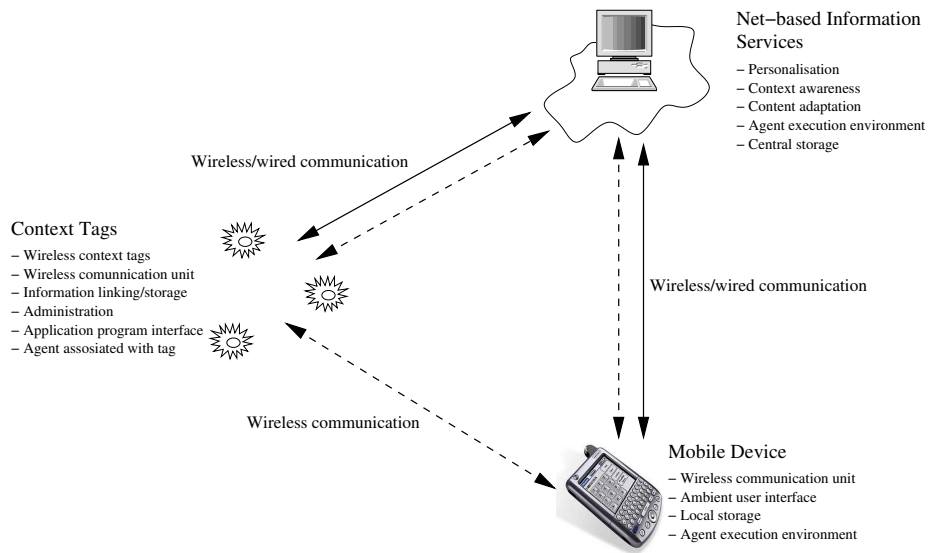
This taxonomy defines five main categories for classifying contextual information (see Figure 1): *i*) Task Context, describing the users activities and goals; *ii*) Social Context, which encompasses information regarding such issues as the user's friends, relatives, and the role which the user occupies; *iii*) Personal Context, which is divided into two sub-categories describing either the physical or mental properties of the user; *iv*) Spatio-Temporal Context, describing attributes such as time, location and movement; *v*) Environmental Context, dealing with the user's surroundings, such as entities. (For a more thorough discussion see Göker and Myrhaug [9])

In this system a top-level taxonomy for context has been defined (see Figure: 1). This ontology is concerned with the types of information that are related to the definition and understanding of context. This context model is not concerned with a complete model from the beginning, but rather imposes a structure that all developers must abide to. The context model is integrated in to a more general domain ontology, which describes concepts of the domain 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 [10], [11].

We argue that this pragmatic approach to an open-ended, yet structured model of context allows application developers to efficiently rule out information that they cannot use as context in their particular application domain.

## 4 Architecture



**Fig. 2.** Overall Architecture

The architecture described in this work was originally developed as part of the AmbieSense<sup>1</sup> research project. The main purpose of this architecture is to supply a generic architecture for implementing context-aware systems in an ambient environment. As part of the AmbieSense project an implementation of this architecture was achieved. The overall architecture for this ambient environment is depicted in Figure: 2, where the three main pillars are: *Context Tags* (small, Bluetooth enabled computers), *Mobile Devices* (PDAs and smart phones) and *Net-Based Information Services*.

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 content, such as a menu from the restaurant where it is mounted, and other available devices in the surroundings.

<sup>1</sup> [www.ambiesense.net](http://www.ambiesense.net)

To enhance the value of their information, Content Providers can offer their content through these Context Tags. This will allow for a personalisation and context-sensitive channel to the end user, where information is disseminated to the mobile user.

The Mobile Device is divided into two main parts: the Context Middleware and the Agency. The Context Middleware offers a general platform for aggregating and storing contextual information from diverse sources. It can furthermore distribute contextual information to other components wishing to utilise context in their service adaption. The primary recipient of contextual information is the agency. The Mobile device typically stores contextual information regarding the user's profile, e.g. goals as part of the task context.

The agency contains all the agents necessary to reason about user situations and handle the advanced services offered through goal identification and problem solving.

This system allows an content provider to offer its services in two different ways. A *stimuli-response* type application can be implemented. This type of application will rely on the built-in capabilities of the Context Middleware to supply information to the user, based on predefined criteria. If this type of application is used, the reasoning agents in the agency will stay dormant. However, it is possible for a content provider to implement an agent and deliver it into the system, thus allowing for far more complex, and possible novel, situations where its information is disseminated to the user.

The rest of this section will describe the two necessary components for applying both simple and complex information dissemination to the mobile device. These components are: the Context Middleware and the agency. For a more thorough description of the AmbieSense architecture see [12].

#### **4.1 Context Middleware**

The Context Middleware considers application such as context aware information services to be *context-aware clients* that utilise contextual information in order to provide the right information to the right time. The contextual information is provided from some context sources. The following sections summarise the core services that the Context Middleware offers to context-aware applications. For a more thorough discussion see [13].

*Storing and retrieving context:* The Context Middleware implements a *context space* in which context is represented and maintained. The context space contains a *current context* and a *context history*. The current context is the relevant contextual information at the present time. When it is 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.

*Generating valid context representations using context templates:* In [13] it was showed how *context templates* can be used to create context patterns that constrain contexts and define contextual information relevant within a particular domain. The context template defines context structures with valid attributes and values. The validation mechanism of the Context Middleware ensures that a given context instance always is valid toward the context template for a domain.

*Context publishing and subscription:* Context-aware clients use subscription mechanisms of the Context Middleware to indicate that they are interested in notifications when context changes as a result of sources publishing new context into the context space. Applications can subscribe to changes in context down to attribute-value level in the context representation (e.g. be notified when location changes).

*Context information linking:* In order to deliver the right information to the right time, context information linking is supported by the Context Middleware. This implies that information entities are explicitly linked to attributes in the context. Retrieving a context from the context history would then also get the information that was relevant for that context.

*Context merging:* Several sources deliver contextual information into the Context Middleware. Input from all these sources need to be merged into one coherent context model. The context merging service allows applications to define what parts of a context that can be merged with similar elements from other contexts. Using context templates and the validation mechanisms, one can verify that two context representations have the same structure and can safely be merged.

## 4.2 Agency

We employed the Jade agent platform [14]. There were two main purposes for choosing this platform. First of all it complies with FIPA, which is the main agent standard today. Adhering to this standard allows for easy implementation of communications between agents in a heterogeneous environment, primarily due to the fact that a standardised Agent Communication Language (ACL) is used.

One of the main parts of this system is the agency which contains the agents that facilitates the complex system adaptation. Beside the Jade specific agents, the agency contains two primary types of agents: the core agents and the application agents.

The core agents are responsible for the basis services associated with the reasoning process. These agents are: the Context Agent, the Creek Agent and the Decomposer Agent. Since this paper is primarily concerned with the use of this system for disseminating information from a information provider perspective, the core agents will only be briefly described.

The Context Agent is responsible for communicating with the Context Middleware. It responds to the contextual events triggered by the Context Middleware, and using the ontology translates them to the representation used inside the agency. It then notifies the agents subscribing to these changes; currently only the Creek Agent.

The Creek Agent utilises Case-Based Reasoning to assess the user's situation. Once a situation has been identified the goals associated with the particular situation is transferred to the Decomposer Agent to be solved. Case-Based reasoning [15] adapts to new situations by remembering similar past experiences (cases). Case-Based Reasoning is a particular promising method for assessing situations in Context-Aware Systems [16], [17], [18]. The version of Case-Based reasoning used and extended in this work is the Creek System [19], [20].

The Decomposer Agent is responsible for satisfying the goals identified by the Creek Agent. This agent uses the Unified Problem-Solving Method description Language (UPML) [21] to decompose the tasks associated with the goal, and to maintain an overview of the Application Agents capable of assisting in the problem solving process.

Each of the Application Agents are responsible for offering their own specific service. These services can range from the mundane, such as offering today's menu, to a complex service of suggesting suitable shops where the user would find interesting and good bargains. Every Application Agent constructed must implement the correct FIPA compliant communication protocol, and supply a specification of its capabilities in UPML.

## 5 An Example – Electronic Patient Journals

This example will discuss the use of the system presented herein, within the context of *electronic patient journals*. This is an area which recently has received a lot of attention. With the introduction of pervasive computing, new and exciting possibilities for supporting the staff emerges. However, in most hospitals mobile applications are not used as an integrated part of the day to day care [22]. This is possibly due to the fact that using mobile applications often is cumbersome, and can easily distract the user with irrelevant or inaccessible information. We suggest that the use of context-aware technology in general, and this system in particular, will reduce the intrusiveness and support the staff in their daily activities.

One possible solution is to rely on adaptive application supplying relevant information at the right time; e.g. only showing the parts of the patient journal that is relevant in the current context, thus making them context-aware (ibid, p. 74). This would ease navigation on the mobile device for hospital staff.

Health care organisations are very complex domains, and for systems to be successful, a minimum requirement is that they are perceived as being usable and efficient by the end users, in this case the health-care staff. We believe that getting the right information at the right time can indeed contribute to make health care applications more usable. This scenario shows how the architecture described in this article can be used to achieve this. First we show a lightweight solution only using the Context Middleware, before a full scale implementation of the system including reasoning agents is described.

Context Tags with the Context Middleware and an agency running are mounted at locations where hospital staff are assumed to have information needs, e.g. in a patient's room. Consider a context model with three attributes, location and time in the *spatio-temporal context*, and colleagues in the *environmental context*. A doctor enters a patient's room where a nurse and the patient is waiting, the mobile device connects to the Context Tag, and the doctor's current context is merged with the tag's current context, which already contains the current location, current time, and the nurse in the colleagues attribute. A search is performed in the tag's context space, and the result is that a matching context is found where `location=(patient's room),time=(10:30)` and `colleagues=(nurse and doctor)`. The matching context is already linked to a

particular part of the patient's journal that is to be filled in when doctor makes her ward round. The link to this information entity is sent to the mobile device of the doctor that uses it to access the patient record server and get the relevant information entity. The relevant part of the patient record is presented to the doctor, ready to be filled in.

To achieve this, small but nevertheless, context-aware application the developer simply needs to construct the correct *context template* and link it with the behaviour required from the system (in this case display the correct journal).

This simple approach can best be characterised as belonging to the *stimuli-response* paradigm. Such an approach works well in horizontal domain where situations are well defined. However, it will not suffice in either vertical domains, or horizontal domains where situations can occur in diverse surroundings. In this example it is likely that at least one of several criteria is not met. Examples could be: the patient has been moved to another room or perhaps even the corridor; a nurse is not present, other staff is present, this particular ward round has been rescheduled to occur at another time. The list of parameters which could change could go on, however the point is that a stimuli-response system must either be very generic, and as a consequence possibly inefficient.

This is one of the main arguments to support this architecture's ability to allow a more flexible approach to context-aware applications. If the developer constructing this particular ward round application wishes to facilitate a more flexible system, he can employ the reasoning capabilities residing in the agency. In this example, this is done by implementing a Ward-Round Agent responsible for the same problem solving behaviour as described above (finding the correct journal), and modelling some prototypical ward round situations to feed into the case base. For a more thorough overview of the reasoning process proposed here and prototypical cases see [23]. With this improved version of the same context-aware application the following scenario is enabled:

As the doctor enters the area where the patient is located a new *current context* is triggered and fed into the agency. This current context is, as in the example above, the merged contexts of the doctor and the Context Tag. In the agency the Creek Agent examines the context and based on other similar situations it concludes that the doctor is doing her ward round, and that she has arrived at this particular patient. This then leads to the goal *facilitate patient journal update*. The Decomposer Agent is notified about this goal, and will decompose it into a task solvable by the Ward-Round Agent<sup>3</sup>. This agent will do the necessary acquisition of relevant information, e.g. journal and test results, and present it to the doctor.

This more complex approach will allow for the same situation (ward round) to occur based on a much wider range of contextual information, thus making the system more flexible and adaptive. The downside is obviously that the developer has to related to more complicated specifications. However, beside the simple interface to the Context Middleware it is sufficient to know the standard FIPA communication protocol; the local ontology used, which is documented within the system; and how to model prototypical cases through the Creek GUI specially designed for novice users. In other words, it is possible to construct complicated adaptive context-aware applications without deep knowledge about the inner workings of this system.

---

<sup>3</sup> Obviously this could be decomposed into a set of tasks, however, to keep this example simple only one task is used.

## 6 Conclusion and further work

We have shown that by utilising our system it is possible to develop both simple and complicated context-aware application through a limited amount of work. It is possible for novice developers to implement a stimuli-response system with a no more knowledge than can reasonably be expected from any developer. It has furthermore been shown how the use of standard protocols and careful ground work allows for development of much more adaptive context-aware system, without any knowledge on the internal workings of the system.

Presently the majority of the AmbieSense system has been implemented and tested at Oslo Airport. The test conducted at Oslo Airport was a usability test, focusing on the travellers response to using context-aware applications. For more information on the test see [13]. However, since the test was focused on the user experience and not on the inner workings of the reasoning mechanism, empirical results on the use of Case-Based Reasoning is as of now unavailable.

We are currently working on applying this system to health care situations, such as ward rounds, where the reasoning capabilities of the system is to be thoroughly tested.

## 7 Acknowledgements

This work was partly funded by the EU commission as the AmbieSense project (IST-2001- 34244).

## References

1. Weiser, M.: Some computer science issues in ubiquitous computing. *Communications of the ACM* **36** (1993)
2. Davies, N., Cheverst, K., Mitchell, K., Friday, A.: Caches in the air: Disseminating tourist information in the guide system. In: *Proceedings of Second IEEE Workshop on Mobile Computing Systems and Applications*. (1999)
3. Oppermann, R., Specht, M.: A context-sensitive nomadic exhibition guide. In: *Proceedings of Second International Symposium on Handheld and Ubiquitous Computing, HUC 2000*. (2000) 127–142
4. Dey, A.K.: *Providing Architectural Support for Building Context-Aware Applications*. PhD thesis, College of Computing, Georgia Institute of Technology (2000)
5. Capra, L., Emmerich, W., Mascolo, C.: Reflective middleware solutions for context-aware applications. In Yonezawa, A., Matsuoka, S., eds.: *Metalevel Architectures and Separation of Crosscutting Concerns - Proc. of Reflection 2001*, Springer Verlag (2001) 126–133
6. Chen, H.L., Finin, T., Joshi, A.: A context broker for building smart meeting rooms. In Schlenoff, C., Uschold, M., eds.: *Proceedings of the Knowledge Representation and Ontology for Autonomous Systems Symposium, 2004 AAAI Spring Symposium*, AAAI Press, Menlo Park, CA (2004) 53–60
7. Dey, A.K.: Understanding and using context. *Personal and Ubiquitous Computing* **5** (2001) 4–7
8. Brézillon, P., Pomerol, J.C.: Contextual knowledge sharing and cooperation in intelligent assistant systems. *Le Travail Humain* **62** (1999) 223–246

9. Göker, A., Myrhaug, H.I.: User context and personalisation. In: Workshop proceedings for the 6<sup>th</sup> European Conference on Case Based Reasoning. (2002)
10. Aamodt, A.: Explanation-driven case-based reasoning. In Wess, S., Althoff, K., Richter, M., eds.: Topics in Case-based reasoning. Springer Verlag (1994) 274–288
11. Jære, M.D., Aamodt, A., Skalle, P.: Representing temporal knowledge for case-based prediction. In: Advances in case-based reasoning. 6th European Conference, ECCBR 2002, Lecture Notes in Artificial Intelligence, 2416, Springer Verlag (2002) 174–188
12. Myrhaug, H.I., Whitehead, N., Göker, A., Fægri, T.E., Lech, T.C.: Ambiesense - a system and reference architecture for personalised context-sensitive information services for mobile users. In Markopoulos, P., Eggen, B., Aarts, E., Crowley, J.L., eds.: Ambient Intelligence: Second European Symposium, EUSAI 2004, Eindhoven, The Netherlands, November 8-11, 2004. Volume 3295 of Lecture Notes in Computer Science., Springer Verlag (2004) 327–338
13. Kofod-Petersen, A., Mikalsen, M.: Context: Representation and reasoning, representing and reasoning about context in a mobile environment. *Revue d'Intelligence Artificielle* **19** (2005) 479–498 To appear.
14. 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, pp. 97-108.
15. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications* **7** (1994) 39–59
16. Zimmermann, A.: Context-awareness in user modelling: Requirements analysis for a case-based reasoning application. In Ashley, K.D., Bridge, D.G., eds.: ICCBR 2003, Case-Based Reasoning Research and Development. Lecture Notes in Artificial Intelligence 2689, Springer Verlag (2003) 718–732
17. Kwon, O.B., Sadeh, N.: Applying case-based reasoning and multi-agent intelligent system to context-aware comparative shopping. *Decision Support Systems* **37** (2004) 199–213
18. Plaza, E., Arcos, J.L.: Context-aware personal information agents. In: Cooperative Information Agents V. Number 2128 in Lecture Notes in Artificial Intelligence. Springer Verlag (2001) 44–55
19. 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 (1991) University Microfilms PUB 92-08460.
20. Aamodt, A.: Knowledge-intensive case-based reasoning in creek. In Funk, P., Calero, P.A.G., eds.: Advances in case-based reasoning, 7th European Conference, ECCBR 2004, Proceedings. (2004) 1–15
21. Fensel, D., Motta, E., Benjamins, V.R., Crubezy, M., Decker, S., Gaspari, M., Groenboom, R., Grosso, W., van Harmelen, F., Musen, M., Plaza, E., Schreiber, G., Studer, R., Wielinga, B.: The unified problem-solving method development language upml. *Knowledge and Information Systems* **5** (2003)
22. Jahnke, J.H., Bychkov, Y., Dahlem, D., Kawasme, L.: Context-aware information services for health care. In Schulz, S., Roth-Berghofer, T., eds.: Modeling and Retrieval of Context 2004 (MRC). Volume 114., CEUR Workshop Proceedings (2004) 73–84
23. Kofod-Petersen, A., Aamodt, A.: Case-based situation assessment in a mobile context-aware system. In Krüger, A., Malaka, R., eds.: Artificial intelligence in Mobile Systems 2003 (AIMS), Universität des Saarlandes (2003) 41–49