

# Context Shadow: An Infrastructure for Context Aware Computing

Martin Jonsson<sup>1</sup>

**Abstract.** In ubiquitous computing environments, hardware and software services will create huge and ubiquitous service spaces. These service spaces will need to be organized in meaningful ways so that the services can be easily accessed and utilized by other services and humans. Context Shadow is a system that makes it possible for services to ask questions about a person's current context, and specifically about what services that are relevant to that context. In the system, sensors and services are organized in meaningful collections, creating a searchable topology of context information.

## 1 INTRODUCTION

Providing information about the users context to applications could be very useful in several ways. The behaviour of an application can be adapted to make the interaction more efficient or to increase the ease of use. You can also imagine entirely new types of applications that are designed specifically to make use of some certain context information. Under the notion of context aware computing there have been several attempts both to define what context really is, and how to design applications that use it [1-2].

In this paper context aware computing will be viewed from a ubiquitous- or disappearing computing perspective [3]. The assumption we make is that our everyday environments will become increasingly more readable and controllable. This is due to the ability to embed computers into everyday objects, and to connect them to the Internet. We also see an increasing use of personal mobile computing artefacts. These new stationary and mobile artefacts will provide different kinds of services either to a user directly or it will be accessible to other software services over the Internet. This results in a potentially infinite space of services that are constantly reachable for humans or other services over the Internet. This kind of ubiquitous service environments and how to deal with them was earlier described in [4]. In the notion of services we also include entities that provide information of the users current context, be it e.g. a location sensor or an application providing information about user activity.

Due to the assumption above we identify one crucial problem, as how to decide what services that are relevant and meaningful in a specific context and how to exclude services that are not interesting. This immediately raises the question of what the

properties are that can be used to identify such meaningful services. One such property, which has been used very frequently in context aware applications, is location, or more precise: proximity [5-6]. It is likely that two services that are close to each other could "interact" in a meaningful way [7-8]. Another property that we see as central is personal association. Many of the services we use have an owner. This might be the software on your personal computing artefacts such as laptops or mobile phones, or it might be some agent-like service residing on a server, maybe collecting information for you on the Internet. Other useful properties could concern social or organizational relations, such as project membership etc.

One way of creating the kind of collections described above is to provide that kind of information through some kind of infrastructure that the different services would share. It has been pointed out that there is a general need for infrastructure to support ubiquitous computing environments [9-10], and specifically to support context aware computing [11].

In this paper we present the Context Shadow system, which provides a way to create searchable clusters of context information.

## 2 CONTEXT SHADOW

The aim of the Context Shadow system is to offer a "shadow of the real world" for applications. By using a simple Java API it is possible to acquire important context information such as information concerning local artefacts, services or people.

More specifically, the system provides:

- 1 Support for context aware service discovery.
- 2 Organisation of services and context information in meaningful collections.
- 3 Context information for applications derived from sensors and other services
- 4 Refinement of context information.

The Context Shadow system is based on a blackboard architecture where certain entities are represented as context servers. These servers serve as repositories for context information related to that entity. Typical entities that can be provided with a context server are persons, locations and groups/projects. A context server contains two types of data: Context information concerning the entity that the server represents and links to other context servers.

Sensors and applications that provide context information are provided with a messaging interface and connected to one or several context servers, where they post their information. The

---

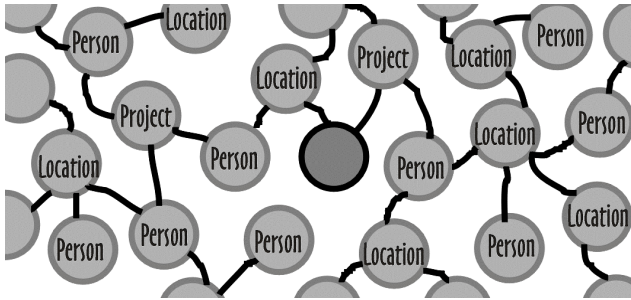
<sup>1</sup> The FUSE Research Group, KTH Center for Wireless Systems, KTH, DSV, Electrum 230, SE-164 40 Kista, Sweden

context servers are implemented using TSpaces from IBM [12]. TSpaces can be described as a network communication buffer with database capabilities implemented as a tuplespace. TSpaces provides a simple and robust network interface as well as advanced querying capabilities.

## 2.1 Cross referencing

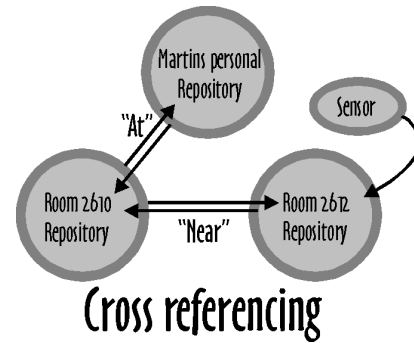
A key feature of the system is the possibility to establish links and relations between different context servers. A typical example of this is the detection of a person entering a room. If either the person detects the room or the room detects the person this results in that a cross reference is established between the local and personal context server. This can be done since the location sensors provide references to a location context server, or in the case of person detection, the person sensor receives references to personal context servers.

The context servers and the links between them create a searchable web where the topology changes dynamically. Information about the users current context does not only consist of chunks of information in the context servers, but is also embedded in the topology of the surrounding web of context servers.



**Figure 1.** The linked context servers create a searchable space, where the topology of the space is part of the context information.

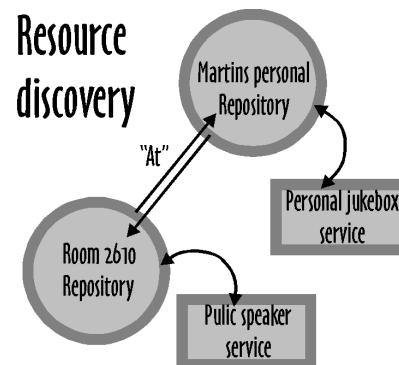
More static references can also be established. Examples of this can for example be references describing the relation between locations. There is a general problem regarding how to represent location in context aware applications. In Context Shadow there is no structured way of describing locations in terms of hierarchies and distances. Instead you define “places” in an arbitrary way, and then create relations between these places. In this way it is possible to create hierarchies when needed, but there is no requirement for developers to provide a complete or coherent location model. Another example of references of a more static nature could be references between persons and projects. By connecting people to each other via a project entity, it is possible to create CSCW tools with knowledge about meeting history, documents etc.



**Figure 2.** The context servers are linked with references. By following the links it is possible to acquire context information from other context servers than your starting point.

## 2.2 Service discovery

The Context Shadow system can be used for different kinds of resource and service discovery tasks. For example, in Fig. 5, a jukebox service is associated with a person. When the person enters a location, the jukebox service will find a public speaker service when it queries the infrastructure for that kind of services. The jukebox service might make queries on a XML description embedded with the speaker service representation, or it can choose to try and match a specific type name describing the service.



**Figure 3.** Using Context Shadow, a jukebox service finds a speaker service at the users current location.

Using the built in functionalities of the underlying TSpaces, you can make complex queries on the data in the context servers. Context Shadow provides an additional API to search the web of context servers created by the cross references described above. The queries can be of the type. “Where am I” or “What other people are in the same location as me” and maybe more useful: “What services of type X are available and relevant to my current context”.

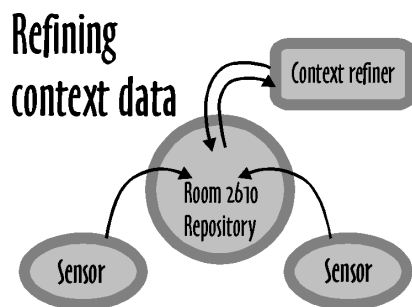
There is also another powerful way to query the infrastructure using XQL (XML Query Language). In Context Shadow you can attach an XML document to every entity, containing various descriptions of the entity. Using XQL you can query the XML description of the entities. This way you can provide a very open interface towards service developers.

In the fuseONE prototype described below, Context Shadow was used partly to discover Jini based services that were relevant at a certain location. In this case the Jini services described themselves by providing a “service ID object”. The querying

service then used the information from Context Shadow to filter out Jini services that were irrelevant to the actual context.

## 2.3 Refinement of context data

Any service should be able to use any data posted into a repository. This raises questions on the format of the data. Different services might want the data in different formats or at different levels of abstraction. Introducing Context Refiners solves this problem. A Context Refiner reads data from the context server then transforms it and adds the new information. The refiners can also be used to handle contradictive data. One example of such a Context Refiner deals with sensor information about a users current location. Since a person only can be at one place a time, location data that differs is contradictive per se. In the implemented refiner each piece of location data comes with a certainty factor and a timestamp. The Context Refiner uses this information to calculate the most probable location and then removes the more improbable location data.



**Figure 4.** A *Context Refiner* reads information from a context server, performs some operation or transformation on the data and then posts the result back to the server.

## 3 EXAMPLE APPLICATIONS

Two prototypes will be described that shows the functionality of the Context Shadow. A messenger service that uses Context Shadow to find public viewer services and a tool for local team work, which uses location information and information about people in the room to provide tools for collaboration.

### 3.1 Messenger service

With this prototype we wanted to examine how public resources can be used to send a message to a person. In the prototype an active messenger service actively tries to find resources near the receiver of the message that can receive the message and display it to the person. The prototype describes following scenario:

1. An active messenger service with a message to a person has migrated to that person's laptop. This computer however lacks output resources that the service can use to display the message.

- 2a. The person enters a room with a public wall display. The messenger service discovers this output resource and chooses to display its message on this resource.

- 2b. The person enters a room where another person sits with his personal computer. This computer has a public speaker resource. The messenger service discovers the speaker resource and chooses to play an audio version of the message through the speakers.

A camera attached to that person's laptop registers the person's entrance in a room. The camera reads and recognizes a barcode-like symbol in the ceiling of the room. The identified symbol is then translated to a reference to the Context Server representing that room, whereby the sensor service on the laptop establishes a cross reference between the context servers representing the room and the person.

The display resource services constantly announce their presence to the Context Server for the room. These services were simple wrappers around a standalone HTML browser and a sound player application.

At this state the active messenger service queries Context Shadow for appropriate display services. First a query is sent that asks for services at the person's current location. This query propagates to other Context Servers using the dynamically established references. In this case it follows the newly established link to the Context Server for the room. If no appropriate services are available there, the query propagates to Context Servers owned by other persons present in the room, thereby also covering services owned by those persons.

### 3.2 Tools for local collaboration

There tend to be more and more artefacts present at meetings. Both personal mobile devices such as laptops and PDA's as well as stationary devices such as projectors. The computer artefacts however often create more problems than is of aid. One irritating obstacle is problems with information flow. A paper you can just reach across the table, but sharing a file on a computer is often more problematic.

We wanted to solve this problem by creating a set of services to enable a seamless flow of information between computers in a room. The underlying system, fuseONE [4] is an attempt to create a ubiquitous service environment, consisting of a large number of standalone components. One set of components consists of simple Jini services that can receive and display documents of different kinds. These services are numerous and reside both on personal and public devices. Another component is a context sensitive service browser, which makes the Jini services described above accessible to users via a GUI. The service browser queries Context Shadow about what Jini services that are relevant to its user's context, then filters out the irrelevant ones from the GUI. More specifically the service browser queries its owners Context Server for (in following order) "personal" services, local services and services owned by local persons.

Whenever a service is found a service description object is returned to the browser application. The application has already found all Jini services in the network used the lookup function that comes with Jini, and uses the service description from Context Shadow to filter out services that are *not* available in the actual room.

The fuseONE system also contains an Active Document service. The Active Document can identify when a certain project has gathered for a meeting, and then actively display information that it has stored from earlier meetings. The service uses information from Context Shadow about project membership, the users locations, what people are in the room and what services that are available in the room.

In this application a person identification sensor was used to create links between location and personal context servers. The actual sensor is an iButton [13], a small button-like computer memory that the users actively have to press against a receiver in the room to announce their presence. The button contains a

reference to the owners Context Server. When the button touches the receiver this reference is being put into the Context Server for the room, and a cross-reference is posted in the personal Context Server.

## 4 RELATED WORK

The problem of creating an infrastructure of meaningful assemblies of services was earlier taken on in the Cooltown project by HP labs [14], where places are provided with a web page. Services that exist in these places can then be accessed via that web page. The system provides several interesting ways to automatically assemble the services at a location and then make them accessible through the web page. The notion of tying services to a location also exists in Context Shadow, with the difference that in Cooltown the ambition is to make services easily accessible directly by users, while Context Shadow has the ambition to provide services mainly to other services.

Another support system, which targets the problem of how to support the design of context aware applications, is the Context Toolkit system from Georgia Tech [1]. This is a middleware system with which you can incorporate sensor data in your applications. The system is built with a widget approach making it possible for applications to incorporate context data about the same way as you incorporate a GUI component. This system does not have an explicit infrastructure approach but rather supports the creation of standalone applications. The Context Toolkit provides much of the same functionality as the Context Shadow, such as context queries and refinement of context data. The major difference from the Context Shadow system is that Context Shadow includes other applications as being part of the context, thus providing support for collaboration between services.

The TEA project presents a system architecture as well as a method to support the design of context aware systems [15]. The system architecture consists of several layers where the bottom layer consists of cues that represent an abstraction of the sensor-data. The cues are then combined into contexts, which can be seen as a high level description of the current situation. These context descriptions can then be fetched by the applications from a tuplespace. The provided method gives step-by-step support for the choice and assembly of sensors as well as for the application development. The architecture from TEA is similar to the Context Toolkit approach in the sense that they both support the development of standalone applications and that none of them uses shared context servers to represent real world entities.

The Interactive Workspace project at Stanford University [16] uses a system they call an *event heap* to enable services in a room to communicate on an event level. The event heap could be compared with the context servers representing locations in the Context Shadow system, with the difference that the context servers are not used for communication between services to the same extent as the event heap. The system also has no support for combining several event heaps into an infrastructure, nor will the event heap communicate with services that have other properties than being in the room.

## 5 FUTURE WORK

One future addition to the system will be to extend the interface to the infrastructure to also handle communication via the SOAP [17] protocol. This would make the infrastructure independent of

programming language, allowing services to post XML-based services to a specific URL.

Another extension concerns the creation of new context servers. At the moment new context servers must be created manually by adding information in a property file. A first step will be to create a web-based user interface that would allow users to create new context servers in an easier way. A more radical change would be to create the ability for context servers to be created in an ad-hoc manner. You could imagine sensors that detects when two persons are close to each other but can provide no information about their physical location. In this scenario you would want to create a temporary location representation only to connect the persons. This context server would disappear after being used.

## 6 CONCLUSIONS

In this paper we have presented the Context Shadow system, which allows applications to make queries about a users context. In the system, services and sensors are tied to Context Servers representing static entities such as persons or locations. These Context servers are then connected to each other in a dynamic way. This novel approach of dealing with context information provides a number of benefits:

Firstly the system simplifies the creation of context sensitive applications, by taking care of issues such as discovery, transport and refinement of context data.

The system creates shared representations of persons and places that can be used by software services. These abstractions are very useful in highly distributed settings, where services might be shared over several machines.

Finally the system scales well. The web-like system structure makes it possible to either distribute the context servers on several machines or to run larger collections of context servers on a single server machine.

## 7 REFERENCES

- [1] A.K. Dey, 'Understanding and Using Context', *Personal and Ubiquitous Computing*, Special issue on Situated Interaction and Ubiquitous Computing, 5(1), (2001)
- [2] B. Schilit, N. Adams and R. Want, 'Context-aware computing applications.' In: First International Workshop on Mobile Computing Systems and Applications, (1994) 85-90.
- [3] M. Weiser, 'The Computer for the 21 st Century.' *Scientific America*, 265(3) (1991) 94-104.
- [4] P. Werle, F. Kilander, M. Jonsson, P. Lönnqvist, C. Jansson, 'A Ubiquitous Service Environment with Active Documents for teamwork support.' Ubicomp 2001 Conference, Atlanta, Georgia, September 2001.
- [5] T. Starner, D. Kirsch and S. Assefa, 'The Locust Swarm: An environmentally-powered, networkless location and messaging system.' The First International Symposium on Wearable Computers, ISWC'97, Boston, USA, October 1997.
- [6] R. José and N. Davies, 'Scalable and Flexible Location-Based Services for Ubiquitous Information Access.' In Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC'99), Karlsruhe, Germany, September 1999. 52-56.
- [7] H. Gustafsson and M. Jonsson, 'Collaborative Services Using Local and Personal Facts.' In: Proceedings from the Personal Computing and Communication Workshop, Lund, Sweden, November 1999.
- [8] T. Pham, G. Schneider and S. Goose, 'Exploiting Location-Based Composite Devices to Support and Facilitate Situated Ubiquitous Computing.' In Proceedings of the 2nd International Symposium on Handheld and Ubiquitous

- Computing (HUC2K), Bristol, UK, September 25-27, 2000. pp. 143-156.
- [9] D. Norman, *The invisible computer*. Cambridge University Press (1999)
- [10] A.C. Huang, B.C. Ling, S. Ponnkanti, A. Fox, 'Pervasive Computing: What Is It Good For?' Workshop on Mobile Data Management (MobiDE) in conjunction with ACM MobiCom '99, Seattle, WA, September 1999.
- [11] A.K. Dey, G. Abowd, D. Salber, 'A Context-Based Infrastructure for Smart Environments.' Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments (MANSE'99), Dublin, Ireland, December 13-14, 1999. pp. 114-128.
- [12] P. Wyckoff, 'Tspaces.' *IBM Systems J.*, Vol.37, No.3, Aug.1998, pp.454-474, <http://www.almaden.ibm.com/cs/TSpaces>.
- [13] iButton is a product of Dallas Semiconductor Corp <<http://www.ibutton.com>>
- [14] D. Caswell, P. Debaty, 'Creating Web Representations for Places.' In Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing (HUC2K), Bristol, UK, September 25-27, 2000. 114-126.
- [15] A. Schmidt, K. Van Laerhoven, 'How to Build Smart Appliances?' *IEEE Personal Communications* 8(4), August 2001. pp. 66-71
- [16] A. Fox, B. Johanson, P. Hanrahan, T. Winograd, 'Integrating Information Appliances into an Interactive Workspace.' *IEEE Computer Graphics & Applications*, Vol. 20, No. 3, May/June 2000.
- [17] Simple Object Access Protocol (SOAP) 1.1, W3C Note 08 May 2000, <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>