Context Based Storage: System for Managing Data in Ubiquitous Computing Environment

Sharat Khungar and Jukka Riekki
Department of Electrical and Information Engineering and Infotech Oulu
P.O.BOX 4500, 90014 University of Oulu, Finland
sharat.khungar@ee.oulu.fi,jukka.riekki@ee.oulu.fi

Abstract - Ubiquitous computing aims to introduce a new concept in computing technology in which the computing infrastructure moves into the background and adapts to the user environment. Context such as location, time and situation allows a system to adapt to the current surroundings in order to facilitate the use of computational environment. In this paper, we describe a novel context based storage system for context-aware ubiquitous computing applications that use context to manage user data and based on the situation make it available to the user. First, we examine several existing systems that use context with documents. Subsequently, a new storage system is presented that uses context to aid in the capture and access of documents in ubiquitous environment. We describe two applications from a set of applications that we have developed with our ubiquitous computing infrastructure and show how they leverage some of the novel features of our storage system to simplify their operations.

Keywords
Context, Context-aware, Data Management, Ubiquitous Computing

1. INTRODUCTION

Context is one of the main factors that differentiate ubiquitous computing from traditional distributed computing. The role of context has recently gained great importance in the field of ubiquitous computing. By context we mean any information about the circumstances, objects, or conditions by which a user is surrounded that is considered relevant to the interaction between the user and the ubiquitous computing environment [1]. The task of making this context information available to components in computer systems has become a prerequisite to advancing human-computer interaction processes in Ubiquitous Computing [2].

Context awareness, or more specifically how to create applications that are context-aware, is a central issue to Ubiquitous Computing research. Context-aware systems are computing systems that provide relevant services and information to user based on his situational conditions [1]. Improved hardware and networking are clearly central in the development of context-aware computing, but an equally important and difficult set of challenges revolves around data management. In order for computing to be invisible to the user while supporting more and more applications, the data required for these applications must be reliably and efficiently stored, queried and delivered.

To address the above issues we have created a Context Based Storage (CBS) system as part of Capnet [3], a middleware architecture that supports building context-aware applications for mobile users. CBS facilitates applications to tag contextual information such as user's activity, location, and time to the documents. The application can use automatic tagging of context or explicitly provide context linked to the specific documents. By using the context, the system provides greater flexibility in storing documents. It then allows applications to retrieve documents by using the context related directly to the document or context related to the user that is then linked to the document with time.

The design and implementation of the storage system presented in this paper contributes to the research of ubiquitous computing by integrating contextual information into the data storage and distributing this information so that it can be accessed on a mobile device, wherever and whenever needed. Novel features of our system include the access rights mechanism for data and support for group activity. The rest of the paper is organized as follows. In section 2 we list related work. In section 3 we describe the context based storage system and its implementation. Section 4 describes the applications we have implemented on top of our storage system. Section 5 ends the paper with some concluding remarks.

2. RELATED WORK

The Forget-me-not [4] was designed for use with a wearable system. It logs the context of the user over time so that it can later be used to find information. The context saved includes information about the user both in the physical and virtual worlds. This system is not directly tied to document storage, but it does allow the context to be used to find accessed files that are stored elsewhere.

The ParcTab [5] allowed access to files that were linked to a particular location. As users move between different locations, the file browser changes to display relevant data. While they only considered location context in their file system, this work was important in establishing the relevance of context in data access. MemoClip [6] consists of a small hardware device, which contains a small database that holds event-location pairs and triggers notifications based on the user's location. CyberMinder [7] provides similar capabilities, but a wide range of context is used, and it can be linked to reminder information, such as time, location, weather, etc. These works have investigated making certain data available in a certain context, but were specific to particular applications. Our aim was to build a generic infrastructure to support a range of applications.

The Stick-e document framework [8] describes information in SGML format that includes data and context information. When a specified context matches an available stored document, a trigger makes data available. However, our approach is different in that we do not use proactive retrieval [9] based on contextual changes in our system, but rather use interactive retrieval to allow the user to choose what action to take. For example, we allow the user to retrieve documents based on context instead of just delivering documents to him when the context matches with the one attached to the document. Gaia CFS [10] bears close resemblance to our system but the focus in Gaia is to simplify locating data important for automatically launched applications in active space. They have put more emphasis on the data adaptation on the device and organising data in file system. In our system special emphasis is given to organizing data with different taxonomies and to support user groups with access rights of data. In addition of supporting context-aware applications for data management, our system also provides data management for different multimedia applications built on the Capnet architecture [15].

The above systems use context about the user, context linked to specific documents and time to provide different ways to store and retrieve documents in the ubiquitous environment. None of the above systems, however, support all three forms of context together with user groups and access rights for data. We have created a Context Based Storage (CBS) that offers the same functionality as the systems discussed above and provides its own additional novel features of data organization with group context and data access rights. We provide attributes linked with documents, to create a storage space along with a facility to log context history. Finally time is integrated throughout the system.

Time Machine Computing (TMC) [11] and Presto [12] systems are designed for use on a desktop computer. TMC helps document storage and retrieval by incorporating time to allow more effective use of the desktop space. Although the desktop representation may not transfer well to mobile environment, this system demonstrates how the context of time can be used effectively with documents. Presto developed a document management system that uses property tags to organize data so that different users may have personalized views of the data hierarchy. It replaces the location based storage system's use of hierarchies and file names with a set of optional properties composed of name-value pairs attached to a document. Any number of properties can be added to data files. This gives the user the flexibility to associate any context desired with the document. Information is later retrieved by searching through the properties. They developed a graphical desktop where files could be grouped together and properties could be dropped on the desktop to display the items that exactly matched the active properties. Our system has some similarities to this system in that our contexts act like their properties. However, their properties act as filters, where each added property restricts the list of data matching the property list. Our work differs from theirs in several aspects. First, the user's context is included in our work to display a relevant document. Second, our retrieval queries are different from filtering. In our system, some environment contexts may not be relevant to a given application, and we therefore ignore such contexts. Finally, we incorporate the mobility of the users, allowing them to access their documents while moving.

In our system, the application can provide context attributes linked to specific documents. These attributes are similar to Presto's properties and relax the naming requirements for documents. Our system also supports the storage of context about the user, similar to Forget-me-not, with context history. The collection of context over time is stored as context history. Context history is used for developing routine learning algorithms [13] in Capnet, by learning the user's behaviour pattern over a period of time.

3. CONTEXT BASED STORAGE

Context-aware computing aims to make the interactions between users and computers easier and supports new interaction styles that are found in ubiquitous computing [2]. Our system uses context to replace many of the tasks that are traditionally performed manually or require additional programming effort. Context is used to constrain the amount of information presented to the user, organize data to simplify locating the data important for users, and retrieve data based on the context of user preference. Applications can provide context attributes linked to specific documents and can retrieve these documents from the storage using these attributes.

CBS consists of a logical context data model and a physical data storage space. The *logical context data model* is a way of representing entities such as people, places, and things. The context information itself is represented using four concepts: entities, attributes, relationships, and groups. *Entities* are simply people, places, and things. Entities represent the base level of context, on top of which more sophisticated representations can be built. Each entity also has access control associated with it, limiting which applications and which people can access its data. *Attributes* describe some property of an entity. For example, people, places, and things all have names. *Relationships* are special kinds of attributes that point to other entities. For example, a person could currently be in a specific place, and this place could contain several things. *Groups* are a way of grouping existing entities, and represent one way in which more sophisticated representations of context can be modelled. For example, an action can be modelled as the person performing the action, the place in which the action happens, and the things used. A work group can be modelled as a collection of the people in that group. A room can be modelled as a place and all of the things in that place.

The *physical data storage* manages how and where the document attached to context data is actually stored. The physical data storage distributes the data so that copies of the context data can exist in multiple places. For example, a person's private context information might reside in his Pocket PC and in his computer at home, while his context information at work might reside in an office computer. This approach makes it easier to scale the system up for large numbers of entities and across wide areas. It also increases robustness to failure by improving the availability of context information. Furthermore, it is more efficient to put the context data close to where it is generated and where it is likely to be used.

The basic structure of the physical data storage is shown in Figure 1.The structure of the data storage comprises (in a logical view), parameters, files, directories, types, entities and groups, and has the following features:

(i) Two kinds of data: parameters (for attributes) and files,
(ii) Parameters and files are owned by entities,
(iii) Entities can be grouped in Groups,
(iv) Groups can have parameters,
(v) The set of parameters and files owned by an entity can be organized into directories and on the basis of their types.

Parameter is a single piece of information related to the user, which is used to store context information with a timestamp. The timestamps are stored with documents, attributes and context history. This allows the system to record when items are created and modified (transaction time) and allows the user to set time according to his needs (valid time). The files and parameters have attributes that specify the owner and rights of access.

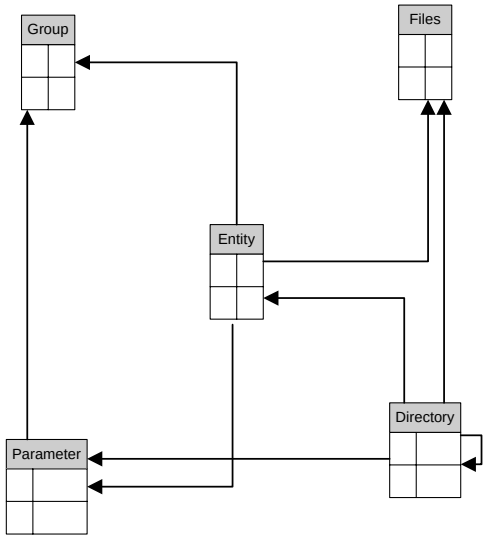

Figure 1 – Entity Relationship diagram for CBS

Files have other attributes that specify the current state (e.g. online, deleted), time of creation and time of last modification. The same user or group of users can have a parameter with multiple timestamps. The files and parameters owned by entities are organized in two basic models: directory and type. *Directory* represents a logical path to the data. The system creates a directory */user* for each user under which its tagged data is stored. *Type* is defined as the data attribute that represents a categorization based on the information of that data. This attribute is automatic: file extension in case of files and context type in case of parameters.

Our system offers access to the data in two modes: Normal and Context-aware mode. *Normal mode* makes all the data available in the user's root directory accessible to him. The other mode of data that is *context-aware* mode is used to organize data by limiting the amount of relevant data according to the context. The system allows files to be tagged with attributes, which may be metadata properties of the media object [15] or with the contextual information defined by the environment. Metadata properties of multimedia documents are also used by multimedia applications in Capnet to retrieve the data they are interested in; context is used to determine in what situation that data is relevant to the user. For example, an application can retrieve data by specifying the format type (e.g. mpeg) of the files it is interested in. Contextual attributes are used by the CBS to

limit the visibility of data according to the context provided by the applications, such as location, activity, time, etc. For example, a file may be tagged with activity context, making it available in a certain situation only.

Attributes are selected based on the requirements of the applications as to which types of attributes are important for tagging with data. Metadata attributes stored with the data are used to specify what type of data is useful for the applications and context is used to specify in which situation that data is made available to the application. Therefore, applications using CBS access data by specifying the metadata properties they are interested in, or by using the context they determine what data is relevant to them and when this data is to be made available. In our system, we have found the following different context information to be most useful for tagging with data: location, identity of the person, activity and time.

The attributes are composed of type and value pair. Files are tagged with specific attributes and moved to context mode from normal mode by inserting a new tuple in the parameter table for the corresponding directory where the file is stored. This way the data stored in the file is tagged with certain context. As files can be inserted from different locations in the same directory, there is a possibility of name clashes. To avoid this clash, all files (also parameters and directories) have a unique identifier index. This identifier is used to distinguish files with the same name in the storage. Parameter identifiers can be used to differentiate user data that is tagged with the same attributes. The reasons for choosing this type of model are: first it is very efficient and flexible for tagging data to different attributes, secondly it allows the applications built on CBS to use the same API for reading the file both in normal and context-aware mode, and thirdly, it allows the different files to be linked to each other without storing them at the same place in one directory.

Data storage of each user is organized in the directory hierarchy structure; different sub-directories are created in the root directory */user* for context mode and normal mode. Data tagged with the contextual information is stored in context mode as in /user/context/ directory and data linked to the metadata properties is stored in as /user/type where type specifies different metadata properties. Although this directory structure is viewed as a hierarchy, context directories impose no fixed ordering, resulting in a forest-like structure. As data can be related to more than one context, our system allows creating different taxonomies of data and using a data thesaurus to find similar structures in the storage.

In the Figure 2, it is possible to see how data can be related using different classification systems. Most of these taxonomies are created automatically and available to the users. This allows the retrieval of data in different and efficient ways. The rectangular nodes define different types of directories and the oval nodes specify data stored in them. For example data node *F2* shown above is an mp3 file which is related to Project1, Project2, music and work where *Project1* and *Project2* are 2 different project sub-directories of the user to whom this file belongs, *music* defines the type of the file and *work* defines the context to which this file is tagged. The *F2* data can be retrieved from the storage by using one or more of the attributes to which it is tagged. For efficient retrieval of documents, data stored in CBS can be linked to other data stored in different directories as well, which helps to interlink relevant data together, e.g. in Figure 2, where the *F2* node is linked to other data nodes i.e. F1 and F3.

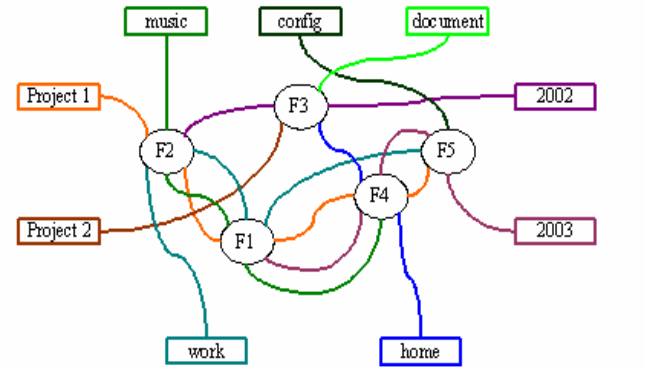

Figure 2 – Data organized using different taxonomies

The activity context in our system is set using the calendar application, while other context used in the system is obtained from physical sensors (e.g. location) or they are implicit (e.g. time) After the data has been correctly tagged, the application accesses the relevant data using CBS based on the contextual information. We separate the attributes from the actual data so that the attributes can be easily searched from the storage with minimal efforts. Documents are retrieved with a query that can contain the document attributes, information about time and context history. For example, retrieve all files for *location == TS387 && activity == meeting && time == 9:30-10:30*. While the above examples illustrates that queries support AND Boolean operations, OR queries are supported by attaching different contexts to the same document.

The access rights mechanism for the user's data deals with the rights to read or write to CBS user files or parameters. It works similarly to UNIX, with three kinds of access zones: owner, groups and others, see Figure 3. Each piece of information is owned by a single user who can belong to several groups (each one with independent access rights). If a user gives access rights to *"others"*, all users can access the data in question. And only the owner can grant and revoke the access rights to his data.

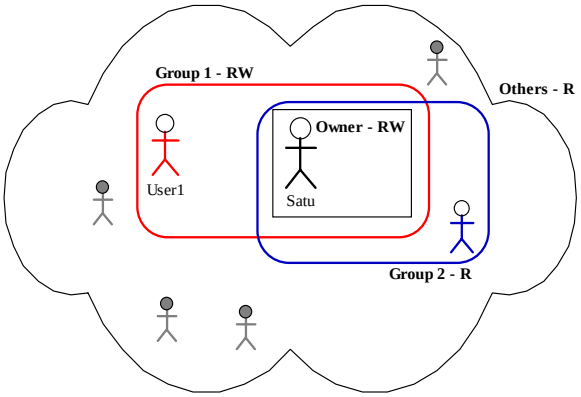

Figure 3 – Access rights example for a particular data object

4. PROTOTYPE

4.1 Capnet System

The Capnet (Context-Aware and Pervasive Networks) program focuses on context-aware mobile technologies for ubiquitous computing. At the highest level, the Capnet architecture is decomposed into Capnet Engines, see Figure 4. Each device that belongs to the Capnet universe contains an engine. An engine may be in a powerful server without any user interface or in a mobile device with many application user interfaces (UIs). As Capnet is a component-based software architecture, the basic building blocks of the engines are component instances, each specialized for producing the functionality of a certain domain area, such as service discovery, user interface, context recognition, media processing, connectivity management, component management, context based storage, or any service added to the system by developers. The Capnet universe is a distributed environment, which means that the engines can use component instances running on the local device as well as remote instances running in engines somewhere in the Capnet universe to provide the required functionality for the applications. An application is composed of application logic and components producing the required functionality. More details about the Capnet system can be found in [3].

CBS resides on one of the engines in the Capnet universe with other components and the physical data storage runs on the server; we assume a continuous connection between the CBS and physical data storage. Applications using the CBS are distributed on the network and use the functionality provided by component management to connect to the CBS. The context component provides abstraction of context information (e.g. location, weather) for its clients in the Capnet universe. All the applications using the CBS obtain the context information from the context component. The context component acts as wrapper for context sensors; it receives sensor data from a number of sensors. Using the sensor data it derives higher-level context information that is utilizable by the other Capnet components including CBS.

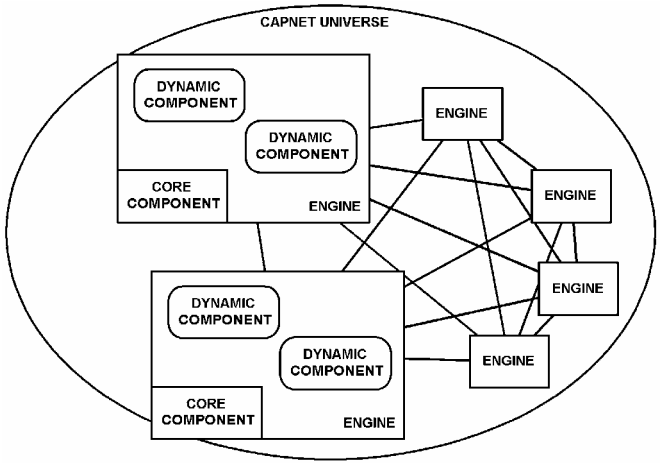

Figure 4 – Capnet Universe

4.2 Applications

We have implemented a set of applications on top of CBS as part of the Capnet system; here we describe only two applications in detail, i.e. file browser (FB) and calendar, which illustrates how context is used to find relevant user data. Data is accessed by an application through an object-oriented interface of the CBS, designed to simplify the retrieval of data types.

FB is a graphical data explorer used to manipulate documents with functions to open, read, write, close, delete and rename. A query interface in FB supports the ability to retrieve documents from the CBS using user's context. The FB operates in two modes: normal mode and context-aware mode. In *normal* mode, all documents accessible to the user are listed. *Context-aware* mode is used for tagging context to documents and copying them to the user's context directory in CBS. In this mode (see Figure 5), the user's documents related to his context provided by the FB are retrieved from storage, and if a query is performed without the context-aware option then all the documents accessible to the user are shown. However, for both normal and context-aware mode, the FB uses the same API provided by the CBS to retrieve data from storage. This API searches for the documents matching the attributes provided by the FB to find the relevant documents from the storage. Unique email-like

identifiers identify users, and an identity is associated with a user when he logs into the system.

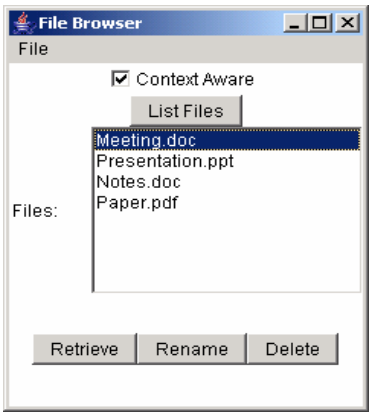

Figure 5 – View from the File Browser application

In context-aware mode, the FB retrieves the user's document from the CBS by using the current context or context attributes tagged with the documents. The FB uses the API provided by the context component [3] to find the current context of the user. The FB uses the location, activity and time information to find the correct files from the CBS. The user can retrieve documents from storage using metadata attributes linked to documents and context history also. Using the API provided by the CBS for retrieval, the FB retrieves the data whose parameter attributes match the context provided by the user. It opens the root directory of the user in the storage and then traverses through the subdirectories to find the matching directory with the attribute used for the retrieval operation.

The user can also change the access rights of his data using the FB; he can either grant or revoke the access rights of his documents to other users of the same group. Using the FB, users can share a document among the other users in a group and tag contextual information also to this document. All the users of the same group can retrieve this document using the FB by performing the search in context-aware mode and this document is listed on the FB if the context of the user matches with the context tagged to this document. This way a document stored at one place can be shared between different users for the same context. The access rights mechanism of the CBS is used for sharing files with different users for same context in a group by granting and revoking access rights for the documents to be shared among the different users of the group. For example, during a meeting the leader of the group creates an "introduction" file for each member of the group attending the meeting. It grants all the members of the group the right to read this file for *meeting* context and certain time duration. Every time that a new member enters the meeting room and performs the retrieval in context-aware mode to list the files related to the current meeting, this "introduction" file is also listed with the other files for the current context.

The activity of the user for a certain duration of time (e.g. meeting) is set via the calendar application, see Figure 6. A document can be attached to this activity and stored in the CBS, which can later be retrieved by the application when the same activity occurs. The user can also access all his documents stored in the CBS by selecting the activities to which they are attached. The CBS opens the sub-directory of the user for that activity and retrieves all the documents stored there. Our system supports tagging different contexts to the same document and organising them according to this contextual information in different directories to associate these documents with different activities. A novel feature of our calendar application is the association of data generated during a special activity (e.g. notes, figures etc) with the activity in the calendar using the context information. Users are then able to review this data by selecting a past entry in the calendar, which would show only the data relevant to the activity.

The user context is based on location, calendar and device profile information. Calendar activity, context information associated with the documents and routine learning algorithms [13] are stored in the CBS. In both applications, the user can also choose where he wants to store a document: in the device memory or in the CBS. This way the user can still access some of his documents from the device memory, when there is no connection between the application and the CBS.

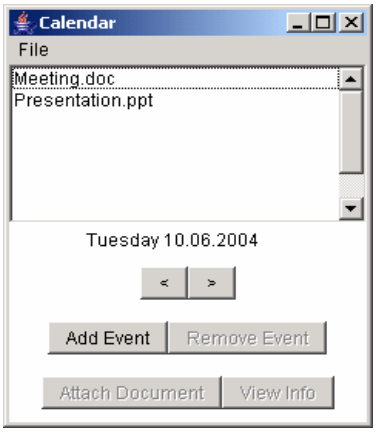

Figure 6 – View from the Calendar application

The CBS also supports the data management of context-aware multimedia applications [15]. An application can capture multimedia content with media components and tag contextual information to it obtained from the context component [3]. The media can then be stored in the CBS and later retrieved by using contextual

information as keywords. Context information is not restricted to location information, as in [16], or a pre-defined set of sources, as in [17], since the context component supports the handling of other types of context, such as meetings. The context-aware instant messaging system [14] developed on the existing Capnet architecture utilises mainly the group activity and access rights mechanism of the CBS.

This prototype supporting different applications has been tested by researchers, and a mobile device was used (Compaq iPAQ PDA) during the experiments. All mobile device components are implemented according to the Personal Java 1.2a specification. Devices are located with the Ekahau positioning engine that utilises WLAN signal strengths measured in the devices. CBS utilises the MySQL database, which is running on a Sun Solaris server. The university's premises covered by WLAN are used as the test environment. The graphical user interfaces are described as extended XUL scripts that are rendered as AWT components in the PDA. All inter-device communications utilise XML-RPC; the open-source Marquee XML-RPC library is used.

5. CONCLUSION

Context information is essential for building applications that support a user's everyday life without causing obstruction in his daily work. Our main goals while developing the CBS were to investigate how context information could be integrated with a data management system, what types of context information would be most suitable to support useful functions in context-aware applications and to provide an architecture for organising data in a ubiquitous environment. The implementation and further experience with the system has helped us to understand better how data management systems must be changed to accommodate the unique characteristics of ubiquitous computing.

We have presented a contextual storage system and two applications built on it that use context to constrain the amount of information presented to the user. Our system supports group activity and provides the user access control on his data. We have found the performance of the system to easily satisfy the requirements of the applications we have developed. For data matching, our system supports AND and OR operations. But instead of Boolean matching, approximate matching using more complex algorithms would be beneficial, most specifically in case of the user searching for documents when the exact context match does not exist.

As we build more context-aware applications, we will be able to determine what types of different applications are best supported in ubiquitous environment. Finally, the need to develop knowledge representations is a major challenge that we are tackling by considering one application scenario at a time.

6. ACKNOWLEDGMENTS

This work was funded by the National Technology Agency of Finland and the Academy of Finland. The authors would like to thank all the personnel in the Capnet program, and the participating companies.

7. REFERENCES

1. Salber, D., Dey, A., Abowd, G.: The Context Toolkit: Aiding the development of context enabled applications. Proc CHI 99 Conference on Human Factors in Computer Systems, Pittsburgh, PA, 1999, pp. 434-441.
2. Weiser, M.: The Computer for the Twenty-First Century. *Scientific American*, September 1991, pp. 94-104.
3. Sauvola, J.: Capnet Project. http://www.mediateam.oulu.fi/projects/capnet/
4. Lamming, M., Flynn M.: Forget-me-not: Intimate Computing in Support of Human Memory, Proc FRIEND21, International Symposium on Next Generation Human Interface, February 1994, pp. 125-128.
5. Schilit, BN., Adams, N., Want, R.: Context-aware computing applications. Proc. IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, 1994, pp. 85-90.
6. Beigl, M.: MemoClip: a location-based remembrance appliance. *Personal Technologies*, Vol. 4, No. 4, 2000, pp. 230–23 4F.
7. Dey, AK., Abowd, GD.: CybreMinder: a context-aware system for supporting reminders. Proc. 2nd International Symposium on Handheld and Ubiquitous Computing, Bristol, UK, September 2000, pp. 172-186.
8. Brown, PJ.: The Stick-e document: a framework for creating context-aware applications. Proc Electronic Publishing 1996, pp. 259-272.
9. Jones, GJF., Brown, PJ.: Context-aware retrieval for pervasive computing environments. Proc First International Conference on Pervasive Computing, Zurich, Switzerland, August 2002, pp. 10-27.
10. Hess, CK., Campbell, RH.: An application of a context-aware file system. *Pervasive Ubiquitous Computing*, Vol. 7, No.6, 2003, pp. 339-352.
11. Rekimoto, J.: Time-Machine Computing: A Time-centric Approach for the Information Environment, Proceedings of the 12th annual ACM symposium on User interface software and technology (UIST' 99), Asheville, NC, November 1999, pp. 45-54.
12. Dourish, P. *et al*., Using Properties for Uniform Interaction in the Presto Document System, Proceedings of the 12th annual ACM symposium on User interface software and

technology (UIST' 99), Asheville, NC, November 1999, pp. 55-64.

13. Pirttikangas, S., Riekki, J., Porspakka, S., Röning, J.: Know Your Whereabouts. 2004 Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'04), San Diego, CA, January 2004.
14. Perttunen, M., Riekki, J.: Inferring Presence in a Context-Aware Instant Messaging System. 2004 IFIP International Conference on Intelligence in Communication Systems (INTELLCOMM 04), Bangkok, Thailand, November 2004.
15. Karunanidhi, A., Doermann, D., Parekh, N., Rautio, V.: Video analysis applications for pervasive environments. Proc. 1st International Conference on Mobile and Ubiquitous Multimedia, Oulu, Finland, 2002, pp. 48-55.
16. Pan, P., Kastner, C., Crow, D., Davenport, G.: M-Studio: An authoring application for context-aware multimedia. Proc ACM Workshops on Multimedia, 2002, pp. 351-354.
17. Keränen, H., Rantakokko, T., Mäntyjärvi, J.: Sharing and presenting multimedia and context information within communities using mobile terminals. IEEE Conference on Multimedia and Expo, Vol.2 2003 , pp. 641-644