Volume 4, Issue 11, November 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Design and Implementation of Mobile Context-Aware Systems Platform for User Application

M. Sasikumar, C. Parthiban, P. Varadarajan
Assistant Professor
Department of Computer Science
Indo-American College, Tamil Nadu, India

Abstract: ***The recent convergence of mobile and context-aware systems has seen a considerable rise in interest in applications that exploit aspects of the operating environment to offer services, tailor application behaviour or trigger adaptation. One area of ubiquitous computing is composed by the context-aware systems, systems where applications are designed to react to the constant changes in the environment. The heterogeneity of the different domains where context is a key parameter has generated different approaches for context acquisition and modelling. Thus, a number of platforms have been developed in order to alleviate the process of application development and to set a common practice for building applications and services. In addition, traditional context-aware applications are poorly suited to highly mobile or distributed environments and often unable to tolerate a rapidly changing execution environment, or take advantage of the availability of new services. Moreover, existing approaches to the development of context-aware applications are, in general, highly reliant upon the underlying infrastructure. Consequently application developers must build their applications with specific environments (indoor or outdoor) or services in mind.***

Keywords: Context Aware Computing, sensing, communications, user mobility

I. INTRODUCTION

With the appearance and expansion of mobile devices, ubiquitous computing is becoming more popular nowadays and the user and his tasks are becoming the focus of application development. One area of ubiquitous computing is composed by the context-aware systems, systems where applications are designed to react to constant changes in the environment. However, the importance of context-based systems extends far beyond systems that are designed around information about the location, user identity, device capability, and services related to this information. Moreover these systems can acquire data for the biomedical functions of people or virtual data coming from software applications, which combined in a new way, can be used in a domain that can improve a person's wellbeing, such as healthcare.

Context awareness, considered as a basic property in the future mobile applications, has gained a momentum in the last few years. The heterogeneity of the different domains where context is a key parameter has generated different approaches for context acquisition and modeling. Thus, a number of platforms have been developed in order to alleviate the process of application development and to set a common practice for building applications and services. They address different architectures, design principles, context representation, sensing approaches, and handling context. Study of the state-of-the-art architectures is an inevitable step for getting better understanding of the problems that application developers face. This, together with the definition of the parameters that describe the context, can lead to

inferring a better way of managing context and making suggestions for extensions of the Context Toolkit, a context-aware platform chosen for analysis, which would increase the functionalities and aid the developers in modeling and supporting context aware applications.

Thus, by examining the structure of the Context Toolkit, this project presents ideas, methods and issues that will lead to a new design of a conceptual model of a context aware platform that will ease the task of adaptive context-aware applications development and will increase its acceptance. With the appearance of mobile devices, ubiquitous systems have gained popularity and application developers have increasingly focused on making applications that target PDAs, mobile phones, notebooks, smart phones etc and the way to use these devices and new technologies to aid the user in performing its daily tasks. This has resulted in rapid integration of these devices in a person's day-to-day life in a manner that the user and his tasks have been placed in the forepart and are central for service development, suppressing the devices, their connectivity and other technical issues.

Existing context-aware applications are, in general, poorly suited to mobile or distributed environments and often unable to tolerate a rapidly changing execution environment, or take advantage of the availability of new services. Moreover, existing approaches to developing context-aware applications are typically dependent on the underlying infrastructure and thus applications are built with particular environments (indoor or outdoor), application domains and infrastructures (computational services or sensors) in mind [Dey,00c]. Consequently, this makes applications less flexible, and difficult to extend or evolve their functionality should the existence of new entities be detected. For example, an application utilising GPS.

When referring to context different context parameters are being given different preference. Context parameters that researches and application developers often list are: location, time, environmental parameters, user activity, device capabilities, identity, network capacity etc. Application developers look into the 'who', 'what', 'where' and 'when' of certain entities and by its analysis they reason about the 'why' of a given occurrence, and program the application logic. However a common way to classify context is by differentiating between physical and logical context.

Physical context is the one that can be measured by hardware sensors, such as: light, temperature, humidity, sound, movement, location etc.

Logical context is the one that is inferred by monitoring the user's behavior, his tasks, his physical and emotional state etc.

II. OVERVIEW OF CONTEXT AWARE FRAMEWORKS

In this section different context aware frameworks are shortly introduced, discussed and compared on common criteria. They adopt different architectural styles mainly driven by the context acquisition, different method of context representation, processing logics and reasoning engines, and at times distinct storage approaches and communication patterns.

The following frameworks will be shortly described:

- Context Toolkit
- Context Broker Architecture
- Context Management Framework
- Gaia

Sensing infrastructure

In order to improve reusability of applications and to alleviate the process of building them, a common practice is to separate the sensing logic from the rest of the system. Hence a common basic module for all architectures is the sensing layer. Context-acquisition begins with sensors. Sensor technology has significantly improved in the past years powered by new solutions that increase quality, reliability, increase the number of parameters that can be measured (temperature, pressure, humidity, acceleration, motion, location, blood pressure etc.) and minimize energy consumption, size and cost. Sensors are sources of contextual data, but the term does not only infer hardware sensors; it encompasses any source of information which provides contextual data and improves the description of a real situation.

COBRA shown in Figure 1 is an agent based architecture that supports context-aware systems in smart spaces (physical places-meeting rooms, homes, vehicles that are equipped with intelligent systems that enable ubiquitous computing). Its central component is the Context Broker, which maintains and manages a shared contextual model on behalf of the collection of agents and is consisted of four main components: Context Knowledge Base, Context Reasoning Engine, Context Acquisition Module and Privacy Management Module.

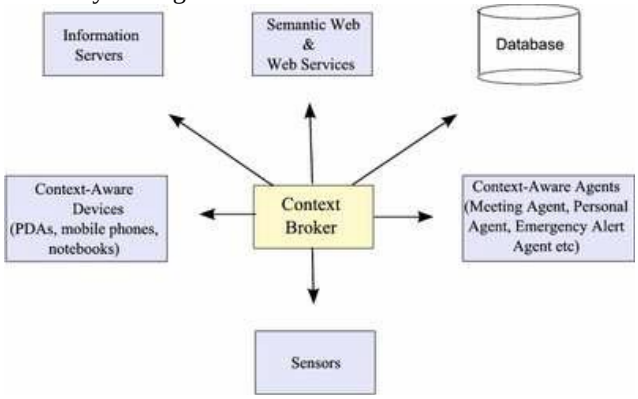

Figure 1: CoBrA Architecture

III. CONTEXT MANAGEMENT FRAMEWORK

The Context Management Framework is shown in Figure 2 and is an example of a framework that adopts the blackboard architectural design. It is consisted of several entities: context manager, resource servers, context recognition services, change detection server, security component and an application.

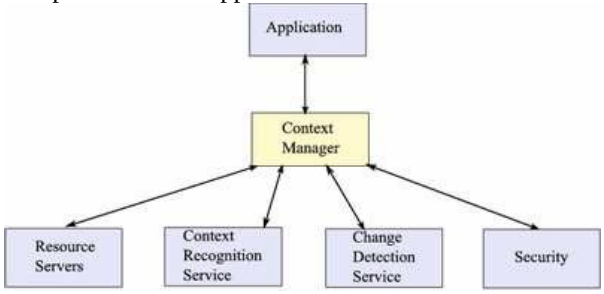

Figure 2: CMF Architecture

The context manager is the central component that manages the blackboard and acts as a central server, it processes the context information acquired from many sources, infers higher level information from them and delivers them to its clients. Data acquisition is performed by the resource servers, while the context recognition services are used on demand by the context manager to deduce complex data out of simple context entities.

What distinguishes this framework from the others, apart from the architectural design, is the advanced way of handling context data represented with ontologies and the usage of fuzzy logic to build higher-level data. However, a drawback of this architecture is that the context manager presents a single point of failure, since application's normal functioning depends directly from it.

IV. ANALYSIS OF THE CONTEXT TOOLKIT

Different designs exist for context-aware systems that depend on several parameters like number of users, location of sensors, device capabilities etc. Context Toolkit encapsulates three different design principles of handling context and building an application which directly refers the manner of acquiring contextual information.

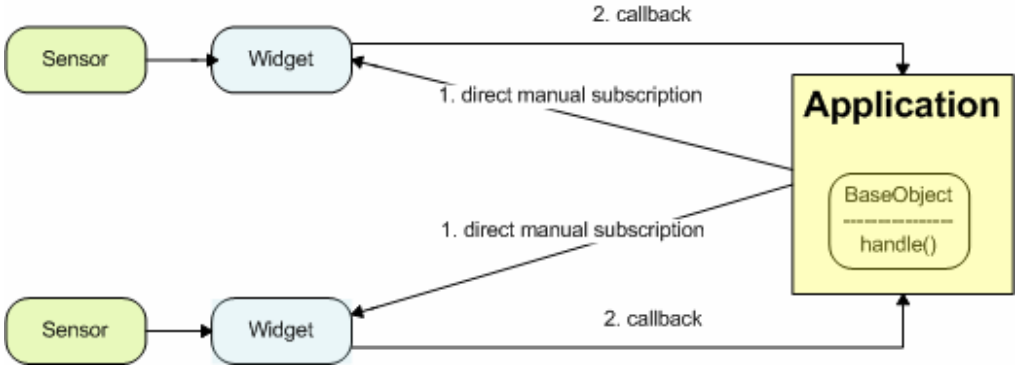

Figure 3: Traditional application design without a discoverer

The relationship between different entities is presented by a configuration that consists of: two widgets, an aggregator, an interpreter, a service, a discoverer and two applications. All components' existence is independent of any application and its communication with other components is automatic and in line with known network protocols. Every component registers its characteristics to the discoverer, alleviating the discovery process for other components that want to subscribe to it. Therefore, aggregators can easily find appropriate widgets, interpreters can locate widgets, and applications can subscribe to suitable widgets, aggregators and interpreters. Communications Infrastructure

One of the requirements set when defining the communications infrastructure of the toolkit has been that the platform supports TCP/IP. This was decided in order to alleviate integration and increase support by many custom built devices, such as: wearable computers, handheld computers, mobile phones and many custom made sensors.

All The BaseObject's default configuration is to support HTTP (Hypertext Transfer Protocol) and XML (Extensible Markup Language). HTTP is used for sending and receiving messages, while XML is used as the language for the data being sent. Its purpose is to facilitate sharing of structured data between different information systems, by defining own elements and to be used on any platform that supports text parsing. XML and HTTP were chosen because they support lightweight integration of distributed components and enable access to heterogeneous

platforms with multiple programming languages. The devices that are used in the architecture should only support ASCII parsing and TCP/IP. There are several flows of actions that need to be closely examined in order to understand the processes that occur within a context-aware application built with the Context Toolkit.

V. CONTEXT-AWARENESS

In order to achieve effective operation in mobile environments, mobile applications must adapt in response to changes in their environment [Davies,94], [Katz,94]. These *adaptive applications* in general require information relating to changes in their surrounding environment, in particular their network quality of service (QoS). To achieve this, aspects of the environment must be sensed or discovered and disseminated among interested parties accordingly. Noble *et al* [Noble,97] describe a taxonomy for adaptation which ranges from placing the responsibility of dealing with adaptation on an individual‟s application (*laissez-faire*) to placing the responsibility for adaptation on the system (*application transparent*). Between these lies *application-aware* adaptation which is based on a collaborative partnership between application and the underlying system. The latter approach allows applications to determine how and when best to adapt, but preserves the ability of the system to monitor resources and enforce allocation decisions. Satyanarayanan *et al* [Satyanarayanan,90] uses Coda as a research vehicle into application-transparency adaptation and more recently have focused on application-aware adaptation with the Odyssey platform [Noble,99]. Adaptive applications have traditionally focused on the monitoring of the network QoS as means for triggering adaptation. The following section describes work in the field of context-aware application design, a field which is more general than mobile adaptive computing since all aspects of the operating environment may be monitored and used in order to trigger application adaptation. Schilit [Schilit,94b] in his work on the PARCTAB system at Xerox PARC [Want,95] describes the context-aware life cycle as *context discovery* (content capture or sensing), *context selection* (interpretation) and *context use*. This broad structure will be used to analyse related work in the area of context-aware computing.

VI. SYSTEM OVERVIEW

A high level system overview of the prototype solution for a ambient home care application is shown on Figure 4. There are several elements that compose the framework: a remote monitoring tool, a set of reusable widgets running independently

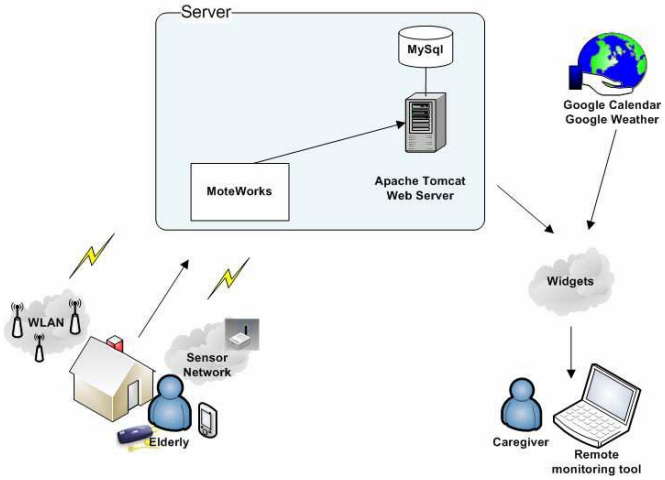

Figure 4: System overview of the prototype solution

from the application, a collection of sensors distributed in the person's home, a set of devices that the person always carries with him (PDA/mobile phone with RFID reader, heart monitor), access to some public web services (Google Calendar and Google Weather), and access to Apache Tomcat web server that encapsulates the processing of some of the data that the widgets deliver to the application. The widgets are run beforehand and independently from the applications. When the remote monitoring tool is started it initializes several aggregators that acquire user specific information and subscribes to the widgets of interest by matching them according to certain criteria (constant and non-constant attributes defined in the widget), which in turn supply the application with notifications about new sensed context data.

VII. CONCLUSION

This paper is focusing on exploring the Context Toolkit, as a sample of a context-aware framework, detecting its strengths and weaknesses, identifying the possible enhancements and successively proposing a design of a new context aware framework. From the analysis performed on the Context Toolkit, it can be concluded that it possesses a number of good design principles such as: reusability, distributed approach, resource discovery mechanism, storage of context data, automatic unsubscribing, context monitoring and feedback etc. However, weaknesses are present as well: the discoverer is a single point of failure, there is a need for clocks synchronization, there is no automatic restarting, there is lack of interoperability support, privacy and security are not yet implemented, there is no reasoning engine, and there is an absence of a conflict solving mechanism.

In this paper, some of these issues were closely looked into and examined. For example, a quality of context mechanism is proposed for improving the quality of the application by making it more reliable and offering service adequate to the application developer's or user's demands. Also several proposals were suggested as a possible way of improving the resource discovery mechanism. Moreover, a matter that would need further focus is the interoperability, in particular making the context data and services provided from one platform available for discovery and usage by others as well. A mechanism has to be developed that will interface with different platforms and will reuse the context sources implemented with different frameworks by providing a common interface that everybody can interact with.

REFERENCES

[1] Bill N. Schilit, Marvin M. Theimer, "Disseminating Active Map Information to Mobile Hosts", IEEE Network, Volume: 8, Issue: 5, September/October 1994, pp. 22- 32, available at: http://impact.asu.edu/~cse591uc/papers/00313011.pdf, last visited: June 2008

[2] B. Schilit., M. Adams, R. Want., "Context Aware Computing Applications", Workshop on Mobile Computing Systems and Applications, December 1994, pp. 85- 90, available at: http://www.ubiq.com/want/papers/parctab-wmc-dec94.pdf, last visited: June 2008

[3] Jesper J. Bisgaard, Morten Heise, Carsten Steffensen, "How is Context and Context-awareness defined and Applied? A survey of Context-awareness", available at: http://www.csconsult.dk/rap/inf7_con.pdf, last visited: June 2008

[4] Bill Schilit, Norman Adams, Roy Want, "Context-Aware Computing Applications", Workshop onMobile Computing Systems and Applications, 1994. Proceedings., Santa Cruz, CA, USA, available at: http://sandbox.xerox.com/want/papers/parctab-wmc-dec94.pdf, last visited: June 2008

[5] J. Pascoe, N. Ryan, D. Morse, "Using While Moving: HCI issues in fieldwork", ACM Transactions on Computer-Human Interaction (TOCHI), Volume 7, Issue 3, September 2000, pp. 417 - 437, available at: http://delivery.acm.org/10.1145/360000/355329/p417-

pascoe.pdf?key1=355329&key2=9722834121&coll=GUIDE&dl=GUIDE&CFID=340 47412&CFTOKEN=41193238, last visited: June 2008

[6] Anind K. Dey, Gregory D.Abowd, "Towards a Better Understanding of Context and Context-Awareness", Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, Vol. 1707, 1999, pp. 304-307, available at: ftp://ftp.gvu.gatech.edu/pub/gvu/tr/1999/99-22.pdf, last visited: June 2008

[7] Terry Winograd, "Architectures for Context", Human Computer Interaction, Volume 15, no. 4, 2000, pp. 263-322, available at: http://hci.stanford.edu/~winograd/papers/context/context.pdf , last visited: June 2008

[8] Oriana Riva, "A Conceptual Model for Structuring Context-Aware Applications", Fourth Berkeley - Helsinki Ph.D. Student, Workshop on Telecommunication Software Architectures, June 2004, available at: http://citeseerx.ist.psu.edu/viewdoc/summary;jsessionid=8199F1E754DF6887BED7B 3223EC8F345?doi=10.1.1.97.3855, last visited: May 2008

[9] H. Chen, T. Finin, A. Joshi, "An ontology for context-aware pervasive computing environments", Special issue on Ontologies for Distributed Systems, Knowledge Engineering Review, Vol.18, No.3, 2004, pp.197-207, available at: http://www.cs.umbc.edu/~finin/papers/ijcai03OntologiesCAPCE.pdf, last visited: June 2008

[10] Matthias Baldauf, "A survey on context-aware systems", Int. J. Ad Hoc and Ubiquitous Computing, Vol. 2, No. 4, 2007, pp. 263 - 277, available at: https://berlin.vitalab.tuwien.ac.at/~florian/papers/ijahuc2007.pdf, last visited: June 2008

[11] Jadwiga Indulska, Peter Sutton, "Location Management in Pervasive Systems" Conferences in Research and Practice in Information Technology Series; Vol. 34, Pages: 143 - 151 , Adelaide, Australia, 2003, available at: http://crpit.com/confpapers/CRPITV21WIndulska.pdf, last visited: June 2008

[12] Wikipedia – Web Ontology Language (OWL), available at: http://en.wikipedia.org/wiki/Web_Ontology_Language, last visited: June 2008

