Simulating Context-Aware Systems based on Personal Devices

Yoosoo Oh, Hyoseok Yoon, and Woontack Woo

***Abstract*— Context-aware systems have been a research focus in Ubiquitous Computing. Advances with regard to context acquisition and activity recognition allow interesting small scale applications. However in larger systems including many sensors and actuators and spanning multiple administrative domains still central issues are not solved. Most systems are designed on a single type of context (e.g. location) or restricted to laboratory size. In this paper we motivate large scale context-aware systems that have support for the full life cycle using a simulation prototype that was implemented and tested. We then describe in detail the underlying architecture. The system is implemented and tested but to further simulate its qualities.**

***Index Terms*— Context-aware system, Context fusion, Context reasoning, Personal Device**

I. INTRODUCTION

Context-aware systems have been a research focus in Ubiquitous Computing since its very beginning [7]. Many systems have been developed in the research community using various sensors and acting on different contexts. However outside the research laboratories context-aware systems have had little impact, with the exception of location aware applications. Nevertheless, many issues with regard to more generic context-aware systems beyond laboratory scale are still unsolved. In particular the life-cycle support of context-aware systems simulation has received little attention. These issues need to be addressed on an architectural level to provide generic solutions. In this paper we will introduce a new and open architecture for context-aware systems that provides mechanisms to solve these problems in parts.

Advances in context-aware computing have been published in the area of acquiring activity and context information using various sensors [2], [8]. This research indicates that for given situations it is in many cases possible to create sensors and recognition subsystems that provide context information to a larger system. It also shows that a system needs to be open and provide appropriate interfaces for different subsystems acquiring context. Looking at the development cycle, it is also of interest to support the simulation of context recognition. As indicated in recent work on prototyping [4], having manually simulated sensor can ease the early phase of development of context-aware systems. Similarly improvement in sensors and recognition algorithms are common and hence a system has to cater for changes on this level.

When the designing and developing of context-aware system and architectures one central issue is to show that the solution presented is appropriate and that it fulfils the criteria set. In the literature several approaches can be found that evaluate the suitability of a solution. Often the argument is made that the architecture was successfully applied to one or more projects, e.g. [1]. This is in our view not necessarily a good measure of the quality as it is from the paper usually not clear how much a specific solution relies on the presented architecture and where there clear benefits in the development and deployment were. The main contributions of the work presented in this paper are a novel architecture for context-aware computing systems and a new approach for simulating context-aware system architectures based on personal devices.

The paper is structured as follows. In section 2 we present a mobile user interface called ubiController. The focus is on simulating physical sensors. In section 3, Context Integrator, the underlying system architecture and context middleware is described. In particular issues regarding context acquisition and context fusion are addressed. In section 4, we present a prototype of a large scale context-aware system. The focus is on including simulated and physical components of various kinds dynamically into a single system supporting multiple. Finally, we report our findings of this simulation for architectures.

II. UBICONTROLLER

We have developed a mobile user interface called ubiController to explicitly allow users to take control and mediate in context-aware computing environments. There are three main features for ubiController, service discovery for universal service control, situation-aware for service filtering and intuitive GUI.

A. The features of ubiController

ubiController is able to discover available services in the network serving as a control point in UPnP architecture [9]. Services are implemented as an UPnP service, so that it can advertise itself through the network. Since UPnP by itself cannot perform communication using context, we incorporate UPnP modules into our context-aware framework, ubi-UCAM (Unified Context-aware Application Model for Ubiquitous Computing), which enables communication using context.

This works was supported by the UCN Project, the MIC 21st Century Frontier R&D Program in Korea, under CTRC at GIST.

The authors are with GIST U-VR Lab., Gwangju 500-712, S. Korea (e-mail: {yoh, hyoon, wwoo}@gist.ac.kr).

Discovered services are directly accessible by clicking service control buttons augmented on the user interface.

To help users to interact more intuitively, ubiController uses a panoramic background of the current environment. As shown in figure 1, a user can see the same scene through the screen of his mobile device where he can click an item of interest to access the service. This interaction metaphor of 'click-and-control' simplifies and helps natural interaction.

B. ubiController as a Simulation Input Device

One ubiController is a representative of one user in our ubiquitous computing scenarios which holds personal information including profiles and preferences of each person. So we can add any number of ubiControllers to represent and simulate the number of people in the scenario.

When a user explicitly gives a command or clicks a button through an interface in ubiController, that information is captured in a form of preliminary context. Preliminary context refers to any information derived from sensors, so by treating users' direct input as such sensor input, we can simulate context-aware systems. For example, ubiController provides an interface to control Light Service. When the user clicks "Turn On" button on the interface, then the information is captured and generated as context to include, who is this input from, what service this input triggers, what service content the input requested, etc.

Figure 1. In this figure a PDA is a Universal Control(ubiController) for simulating sensors.

III. A NEW ARCHITECTURE FOR CONTEXT INTEGRATION

Different architectures and middleware systems to support context-aware systems have been developed over recent years, examples from different domains include the Context-Toolkit [5], the TEA context acquisition architecture [6], and the JCAF middleware [1]. Even though there have been advances, especially improving the separation of concerns, many issues, in particular with regard to large scale context-aware systems are still unsolved. Central concerns for simulating context-aware system need to be addressed on an architectural level to provide generic solutions. In this section, we introduce a novel architecture for context fusion and reasoning designed for large scale context-aware systems.

A. Context Integrator Architecture

Context fusion and reasoning are the central functionality provided by the Context Integrator, which is a novel architecture for context-aware systems. An architectural overview is shown in Figure 2.

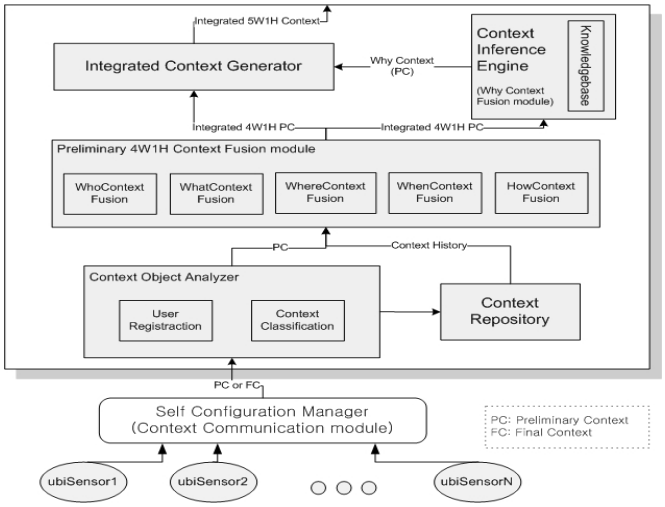

Figure 2. The architectural of the Context Integrator.

The Context Object Analyzer collects contexts (preliminary or final alike) periodically from various kinds of sensors which are placed semantically in the same active area. The Context Repository stores and manages context the history of the integrated contexts. The Preliminary Context Fusion module integrates the inputted Preliminary Contexts as an integrated 4W1H context according to characteristics of each sub-context of 4W1H (Who, What, Where, When, and How). Integrated Contexts are created by applying the appropriate fusion method. Preliminary Context (PC) expresses the feature information and sensor description. Integrated Context (IC) forms the complete 5W1H context generated by Context Integrator which additionally includes the inferred intention using the context inference engine. The Context Inference Engine has reasons the 'Why' component of the context (intention) by using the result of Preliminary Context Fusion module. It contains a knowledgebase which consists of facts and rules that describe the behavior of the system. This is described in more detail in the next section. The Integrated Context Generator combines all components into a 5W1H Integrated Context. This contains information, such as user's identity, location, activities, behavior, patterns, and intention.

The generic behavior and functionality of a specific context-aware system running based on Context Integrator are specified by rules. Changes and reconfiguration of the system by a developer or maintainer are done by adding, deleting, and modifying rules. For simplicity these rules are specified in plain text and reconfiguration just requires a text editor. Hence changes to the overall system behavior do not require changes in any source code in Context Integrator. In the implementation of the Context Integrator the Context Inference Engine is implemented on top of JESS (Java Expert System Shell). Rules

are specified in syntax similar to LISP. Compared to declarative programming language this eases reconfiguration as rules do not require order.

Throughout the development of a middleware which implement the architecture described above tests based cases were performed. These test cases were based on simulated sensors (ubiController) integrated with the simulation environment of the virtual building implemented by Macromedia Flash.

B. *Explanation and Accountability*

To make it feasible for developers and operators to understand how and why a system is performing certain actions, a component for explanation was added. In the simulation example this was also implemented in Flash to allow easy changes and provide more options for visualization. By logging the information presented in the explanation tool we hope to achieve accountability for actions taken on context. In a real environment it is essential to know why the heating is switched off or why certain people have access to a certain area.

During the development of prototypes it proved to be an invaluable tool for debugging. Based on the information provided, it was in many cases possible to quickly find out where the problem was. Often in larger context-aware system, finding the component responsible for the systems misbehavior is a major step in finding an error.

The simulation example can show the intention output message from Context Integrator for debugging. This message describes the user's current intention with his current activity as a point of view of an outsider. For easy debugging to developers, the significant information (eg. the weighting factor to the current output) is presented and also the previous information (activity, location, no of persons) is revealed.

IV. Simulation for a Large Scale Context-Aware System

To explore issues related to large scale context-aware systems we implemented an example system that includes different features that we see as important for in this area. These requirements have been deduced from a number of projects which are described in literature [3] or have been conducted by the authors. To assess these requirements we included them into one prototypical system. To assess these requirements we included them into one prototypical system. In particular the focus is on:

- The system should support context-awareness for large systems which contain many sensors and actuators.
- Sensors, actuators and computational components should be independent of each other (and it is likely that they are under different administrative control).
- Context, derived at different units within the system, should be maintained.
- The system should support online dynamic changes. It is assumed that components like sensors, actuators or computational components can be added, replaced, or broken.
- The system should provide means for accountability and explanations for actions taken automatically.

The prototypical system was implemented and some parts are shown in Figure 3. The prototype is a context-aware building including a number of real sensors(ubiControllers) as well as a number of simulated actuators(ubiBuilding). Having simulation as basic concept allows development before a real environment can exist. This is in particular interesting when developing context aware systems for new environments that do not exist when the context-aware software is developed but where the software is required when the environment is ready. For example, while a new building is physically built the software can already be developed and tested in simulation so that when the building is finished the tested software can be put to use. This is significant for buildings where context-awareness is at the center of controlling access and heating/ventilation of a building.

Figure 3. In this figure components of the prototypical system are shown. In an output simulator for a large building implemented in Flash is depicted (ubiBuilding).

The real input components have been implemented as typical representative of sensors we expect in a large scale context-aware system. The real output component can be any component in a real building system which can be controlled electronically, e.g. door locks, heating, shutters or lightning. In the prototype these are not implemented yet.

The simulated output component is ubiBuilding Simulator which is a building simulation system implemented in Macromedia Flash, partly shown in Figure 2 a). It simulates a full scale building comprises 5 floors of 19 rooms with in total of 41 actuators (9 door-lock, 19 lights, and 13 heaters).

The central process of the system is context fusion module and computation engine. Context fusion module integrates contexts obtained from all sensors attached to the system. The computation engine is an inference rule engine based on JESS (Java Expert System Shell). The rules are fired by user's

explicit or implicit input which is behavior rules, event rules, and explicit or implicit control rules. This central component is described in the next section in more detail.

Implementing a prototypical system, which combines many features we expect in context-aware systems, along with the development of the context integration architecture helped for testing. Having a set of applications, based on real and virtual sensors makes it easy to create test cases so that basic principles of software testing can be applied during the development.

V. CONCLUSION

In this paper we have introduced Context Integrator, a novel architecture for context-aware systems. The proposed architecture has been used to develop prototypical systems. Its central component is a rule based system that creates a system behavior based on contexts detected. Using rules as the basic means are a flexible way to change and manipulate the system behavior during all phases of the life cycle of a context-aware system. The central advantage for large scale systems is that a change in behavior can be introduced at run time by adding, changing or deleting rules.

In a prototypical system we showed that given this level of abstraction it is easily feasible to have components exchangeable including personal devices in the system, for sensors and actuators alike. Especially for the development of context-aware software-systems in parallel to the building of the real environments, simulation of sensors and actuators becomes very important.

To make the context-aware system understandable, in a first step for the developer and operator and later to the user, a component that provides explanations for decisions taken is introduced. Providing information why certain actions are taken and based on what data intentions have been inferred is crucial for developing understandable and user-friendly context-aware systems.

Currently we further develop the Context-Integrator architecture to fix issues those surfaces in the expert evaluation. Additionally we assess further projects on large scale context-aware systems where the developed system can be applied. With regard to the simulation for context architectures we are planning to use the method in further related projects such as the assessment of toolkit support.

REFERENCES

[1] Jakob E. Bardram. The Java Context Awareness Framework (JCAF) - A Service Infrastructure and Programming Framework for Context-Aware Applications. In Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive 2005), Lecture Notes in Computer Science, Munich, Germany, May 2005. Springer Verlag.

[2] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Proceedings of PERVASIVE 2004*, vol. LNCS 3001, Springer-Verlag, 2004, pp. 1-17.

[3] G. Chen, and D. Kotz. *A Survey of Context-Aware Mobile Computing Research*. Dartmouth College, Technical Report: TR2000-381, 2004.

[4] Steven Dow, Blair MacIntyre, Jaemin Lee, Christopher Oezbek, Jay David Bolter, Maribeth Gandy, "Wizard of Oz Support throughout an Iterative Design Process," *IEEE Pervasive Computing*, vol. 4, no. 4, pp. 18-26, Oct-Dec, 2005.

[5] D. Salber, A.K. Dey and G.D. Abowd, "The Context Toolkit: Aiding the Development of Context-Aware Applications," *In the Workshop on Software Engineering for Wearable and Pervasive Computing*, Jun. 2000.

[6] A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven, and W. Van de Velde. Advanced interaction in context. In H.W. Gellersen, editor, Proc. of First International Symposium on Handheld and Ubiquitous Computing (HUC99), volume 1707 of LNCS, pages 89--101. Springer-Verlag, 1999.

[7] B. Schilit, N. Adams, and R. Want, "*Context -Aware Computing Applications*," Proc. Workshop Mobile Computer Systems and Applications, IEEE CS Press, Los Alamitos, Calif., 1994, pp. 85-90.

[8] K. Van Laerhoven, A. Schmidt and H.-W. Gellersen. "Multi-Sensor Context-Aware Clothing". In Proc. of the 6th Int. Symposium on Wearable Computers, ISWC 2002, Seattle, WA. ISBN: 0-7695-1816-8; IEEE Press, 2002, pp. 49-57.

[9] Universal Plug and Play (UPnP), http://www.upnp.org