Human Factor Issues in Building Middleware for Pervasive Computing

Tatsuo Nakajima, Eiji Tokunaga, Hiroo Ishikawa, Kaori Fujinami, Shuichi Oikawa
Department of Computer Science
Waseda University
tatsuo@dcl.info.waseda.ac.jp

Abstract

Our daily lives will be dramatically changed by embedded small computers in our environments. The environments are called *pervasive computing environments*. To realize the environments, it is important to reduce the cost to develop pervasive computing applications by encapsulating complexities in middleware infrastructures that are shared by various applications.

In this paper, we present three middleware infrastructures for pervasive computing, that have been developed in our projects. These middleware infrastructures hide various complexities such as context-awareness and distribution to make it easy to develop pervasive computing applications. We also describe some design issues related to human factors that we found while designing and implementing our middleware infrastructures.

1 Introduction

Our future daily lives will be augmented by various computers and sensors, and our environments change their behavior according to the current situation. Each person will have many personal devices such as cellular phones, PDAs, MP3 players, voice recorders. Also, many appliances will be available near us such as various displays, televisions, and information kiosk. We believe that these devices and appliances are communicated in a spontaneous way, and provide useful information to us.

Therefore, our daily lives become more and more complex every day. Information technologies have been increasing these complexities, because a large proportion of our daily lives is currently spent in analyzing various sorts of information. Ironically, present pervasive computing technologies will increase the amount of such information dramatically, and increase the complexities in our daily lives. Also, various appliances surrounding us rapidly become commodities. Today, it is very difficult to create an appliance that offers special, distinctive features. For example, we cannot distinguish among different vendor's televisions. Therefore, it is important to take into account pleasurable experiences when a user uses the appliances[12, 3].

It is important to offer middleware infrastructures to hide a variety of complexities from application programmers to make it easy to develop pervasive computing applications[14, 6]. However, we have not enough experiences how to build middleware for pervasive computing, and it is important to share knowledge among research communities. In this paper, we present three middleware infrastructures for pervasive computing, that have been developed in our projects. These middleware infrastructures hide various complexities such as context-awareness to make it easy to develop pervasive computing applications. We found that hiding context-awareness requires to take into account human factors when designing middleware. We show the overview of our middlewares, and discuss some human factor issues while designing our middlewares.

The remainder of this paper is structured as follows. In Section 2, we describe our vision called computer mediated daily lives. Section 3 presents middleware for pervasive computing. In Section 4, we show three middleware infrastructures that we have developed in our project. Section 5 shows research agenda for attacking the issues, and Section 6 concludes the paper.

2 Computer Mediated Daily Lives

As described in the previous section, our future daily lives will be mediated by various computers that offer various interaction devices and sensors. We call the future life style *computer mediated daily lives*. Computers will connect a variety of spaces such as home, office, town, car, train, bus, station, and airport in a seamless way. When we develop application software in computer mediated daily lives, it is important to design the behavior of human to reduce the complexities in our lives and to increase pleasurable experiences. However, currently, we do not have a common software infrastructure to develop these applications in an easy way.

The purpose of a software infrastructure for computer mediated daily lives is to retrieve information from our real world that could not be made available before, and to control various everyday objects that could not be controlled before by embedding computers. One of the most important issues in computer mediated daily lives is to provide context-awareness, to integrate physical and cyber spaces to personalize our real world and reduce the complexities in our dairy lives[1].

Such a research shows that a software infrastructure to support computer mediated daily lives is a key to realizing the vision. The infrastructure makes it possible to share various devices and sensors, and to build pervasive computing applications easily. We have worked with many companies in building embedded systems, and have found that the following traditional following choices are wrong.

- A future appliance will be extremely smart.
- Our environment will be extremely smart.
- Each appliance will have a well defined fixed interface.

The first choice means that each application contains many functionalities to satisfy various requirements. For example, current digital televisions are very complex and include various functionalities to satisfy most of us. However, the cost of the television is inflated, and it is not easy to extend to support future advanced services. Similarly, the second choice requires very high cost to build smart environments, as shown in much research. The most serious problem with the choice is that it is not easy to support collaboration with environments and appliances that are carried by a user, due to special protocols that are assumed by the environments. The last choice cannot allow us to upgrade each appliance's service independently.

A key to solving the problems is provide middleware infrastructures for integrating a variety of services provided by both appliances and environments. The middleware infrastructures offer high level abstraction that hide various complexities likes context-awareness, distribution, and heterogeneity from application programmers. In our project, we have developed several middleware infrastructures for pervasive computing[4, 5, 7, 8, 10, 13]. In this paper, we present three middleware infrastructures that hide context-awareness to reduce complexities when developing pervasive computing applications.

3 Middleware for Pervasive Computing

This section describes three middleware infrastructures that have developed in our project. The first middleware infrastructure makes it easy to develop mixed reality applications for pervasive computing. The middleware hides distribution and automatic configuration according to changes of a user's current situation. Also, it is easy to build an application by composing several multimedia components. The second middleware infrastructure allows us to use various interaction devices to navigate traditional GUI-based applications. The middleware hides to select the most suitable interaction devices to control appliances. The third middleware infrastructure provide an integrate way to control home appliances. The middleware hides the personalization to control home appliances.

3.1 Middleware and Abstraction

There are a variety of complex issues in pervasive computing environments. For example, there may be various devices that have dramatically different characteristics. These ultra heterogeneity should be hidden in middleware infrastructures. Also, pervasive computing applications need to control many devices that are connected via networks, but applications may not take into account such distribution. Also, the structure of applications should be dynamically reconfigured according to the current situation. However, these complexities should be hidden from an application programmer to make it easy to develop their applications.

The behavior of pervasive computing applications should be changed according to the current situation. For example, the behavior may be personalized according to a user's preference. Also, the most suitable devices to be used by an application may be changed according to a user's current location. We believe that it is important to hide the changes under the abstraction provided by middleware to make the development of applications easy. Therefore, application programmers need not to take into account these changes.

3.2 Middleware for Mixed Reality

3.2.1 Overview

Our middleware infrastructure called MiRAGe[13] consists of the multimedia framework, the communication infrastructure and the application composer. The multimedia framework, is a CORBA-based component framework for processing continuous media streams. The framework defines CORBA interfaces to configure multimedia components and connections among the components.

Multimedia components supporting mixed reality can be created from the MR class library. The library contains several classes that are useful to build mixed reality applications. By composing several instances of the classes, mixed reality multimedia components can be constructed without taking into account various complex algorithms realizing mixed reality.

The communication infrastructure based on CORBA consists of the situation trader and OASiS. The situation trader is a CORBA service that supports automatic reconfiguration, and is colocated with an application program. Its role is to manage the configuration of connections among multimedia components when the current situation is changed. OASiS is a context information database that gathers context information such as location information about objects from sensors. Also, in our framework, OASiS behaves like as a Naming and Trading service to store objects references. The situation trader communicates with OASiS to detect changes in the current situation.

Finally, the application composer, written by an application programmer, coordinates an entire application. A programmer needs to create several multimedia components and connect these components. Also, he specifies a policy on how to reconfigure these components to reflect situation changes. By using our framework, the programmer does not need to be concerned with detailed algorithms for processing media streams because these algorithms can be encapsulated in existing reusable multimedia components. Also, distribution is hidden by our CORBA-based communication infrastructure, and automatic reconfiguration is hidden by the situation trader service. Therefore, developing mixed reality applications becomes dramatically easy by using our framework.

3.2.2 How Our System Works?

In a typical mobile mixed reality application, our real-world is augmented with virtual information. For example, a door of a classroom might have a visual tag attached to it. If a PDA or a cellular phone, equipped with a camera and an application program for capturing visual tags, the tags are superimposed by a schedule of today's lecture.

We assume that in the future our environment will deploy many mixed reality servers. In the example, the nearest server stores information about today's lecture schedule and provides a service for detecting visual tags and superimposing them by the information about the schedule. Other mixed reality servers, located on a street, might contain information about what shops or restaurants can be found on the street and until how late they are open.

To build the application, an application composer uses components for capturing video data, detecting visual markers, superimposing information on video frames and displaying them. The application composer contacts a situation trader service to retrieve a reference to a reference to the nearest mixed reality server to a user. When he moves, a location sensor component notifies sensed location information to OASiS, and OASiS notifies the situation trader to replace the current object reference to the reference of the nearest mixed reality server. In this way, the nearest mixed reality server can be selected dynamically according to his location, but the automatic reconfiguration is hidden from an application programmer.

3.3 Middleware for Interaction Devices

3.3.1 Overview

In the middleware infrastructure[4], an application generates bitmap images containing information such as control panels, photo images and video images. Also, these applications can receive keyboard and mouse events to be controlled. The user interface middleware receives bitmap images from applications and transmits keyboard and mouse events. The role of the middleware is to select appropriate interaction devices by using context information. Also, input/output events are converted according to the characteristics of interaction devices.

The application implements graphical user interface by using a traditional user interface system such as the X window system. The protocol between the middleware and the user interface system are specified as a standard protocol. In the paper, we call the protocol the universal interaction protocol.

Our system consists of the following four components.

- Interactive Application
- UniInt Server
- UniInt Proxy
- Input/Output Interaction Devices

Interactive applications generate graphical user interface written by using traditional GUI toolkits. In our system, we can use any existing GUI based interaction applications, and they are controlled by various interaction devices that are suitable for the current situation.

The UniInt server transmits bitmap images generated by a window system using the universal interaction protocol to a UniInt proxy. Also, it forwards mouse and keyboard events received from a UniInt proxy to the window system. In our current implementation, we need not to modify existing servers of thin-client systems, and any applications running on window systems supporting a UniInt server can be controlled in our system without modifying them.

The UniInt proxy is the most important component in our system. The UniInt proxy converts bitmap images received from a UniInt server according to the characteristics of output devices. Also, the UniInt proxy converts events received from input devices to mouse or keyboard events that are compliant to the universal interaction protocol. The UniInt proxy chooses a currently appropriate input and output interaction devices for controlling appliances. To convert interaction events according to the characteristics of interaction devices, the selected input device transmits an input specification, and the selected output device transmits an output specification to the UniInt proxy. These specifications contain information that allow a UniInt proxy to convert input and output events.

The last component is input and output interaction devices. An input device supports the interaction with a user. The role of an input device is to deliver commands issued by a user to control home appliances. An output device has a display device to show graphical user interface to control appliances.

In our approach, the UniInt proxy plays a role to deal with the heterogeneity of interaction devices. Also, it can switch interaction devices according to a user's situation or preference. This makes it possible to personalize the interaction between a user and appliances.

3.3.2 How Our System Works ?

An example described in this section is a ubiquitous video phone that enables us to use a video phone in various ways. In this example, a user speaks with his friend by using a telephone like a broadband phone developed by AT&T Laboratories, Cambridge. The phone has a receiver like traditional phones, but it also has a small display. When the phone is used as a video phone, the small display renders video streams transmitted from other phones. The display is also able to show various information, such as photos, pictures, and HTML documents that are shared by speakers. Our user interface system makes the phone more attractive, and we believe that the extension is a useful application in pervasive computing environments.

When a user needs to start to make a dinner, he will go to his kitchen, but he likes to keep to talk with his friend. However, a phone is put in a living room, and the traditional phone receiver is not appropriate to continue the conversation with his friend in the kitchen because his both hands may be busy. In this case, we use a microphone and a speaker in the kitchen so that he can use both hands for making the dinner while talking with his friend. In the future, various home appliances such as a refrigerator and a microwave provide displays. Also, a kitchen table may have a display to show a recipe. These displays can be used by the video phone to show a video stream. In a similar way, a video phone can use various interaction devices for interacting with a user. The approach enables us to use a telephone in a more seamless way.

Our system allows us to use a standard VoIP application running on Linux, which does not take into account using sophisticated interaction devices. The application provides a graphical user interface on the X window system. However, our system allows a user to be able to choose various interaction styles according to his situation. Also, if his situation is also changed, the current interaction style is changed according to his preference.

3.4 Middleware for Home Computing

3.4.1 Overview

A personal home server[10] is implemented in a personal devices like a cellular phone, a wrist watch, or a jacket. Thus, the server can be carried by a user anytime anywhere. The personal home server collects information about home appliances near a user, and creates a database storing information about these appliances. Then, it creates a presentation document containing the attributes of appliances and the commands to control them. A display near the user also detects the personal home server, and retrieves the presentation document containing the automatically generated user interface. The display shows the presentation document on the display. The document contains URLs embedding the attributes of appliances and their commands. Also, the presentation document is customized according to a user's preference. A personal server contains preference rules to represent a preferences for the owner of the personal home server. The execution of the rules is hidden from an application. When a user touches the display, an URL containing the attributes of an appliance and its command is transmitted to his personal home server via the HTTP protocol. The server translates the URL to a SOAP command by using a database containing information about the appliance that he likes to control. Finally, the SOAP command is forwarded to the target appliance.

3.4.2 How Our System Works?

In this section, we describe how our systems works by using an example. In the example, we consider that a television and two lights are in a room. We assume that one is a ceiling light and another is a floor light. When a user enters a room, his personal server in his pocket detects appliances by receiving SSDP advertisement messages containing IP addresses from the television and the lights. The personal home server accesses Web servers of both appliances, and retrieves their service description documents. Then, the documents are stored in the service database of a personal home server.

On the other hand, a display device near the user detects an advertisement message from the personal home server, and retrieves an automatically generated HTML document. The document contains URLs for controlling the television and a ceiling light. The user specifies that he likes to use only ceiling lights in a rule stored in the context database. Thus, the context management module removes information about a floor light.

The document is shown by a browser of the display device. In this example, a user can navigate the browser by using a touch panel. When the user clicks an URL, http://102.10.2.2/?func=light&?type=ceiling&!power=on, a GET command containing the URL is transmitted to the personal home server. The server searches a light appliance that matches to specified attributes in the service database, and finds that its IP address is 102.10.2.10. Also, a command encoded in the URL is converted to a SOAP request, power("ON"). Finally, the commands is transmitted to a target light appliance whose IP address is 102.10.2.10, then the appliance turns on its power. In our current prototype, light appliances are controlled via an X10 device, and an analog television is controlled through an intelligent remote control device.

4 Human Factor Issues for Pervasive Computing Middleware

In our middleware infrastructures described in the paper, context-awareness is hidden from a user. However, these properties are closely related to human factors. For example, if an interaction device is switched in an unexpected way, a user may surprise the context change. This means that middleware infrastructures that hide context-awareness or dynamic reconfiguration will need to take into account human factors when designing the middleware[2]. For example, our experiences show that the automatic selection of interaction devices is not good approach. Instead, we use a token to choose the most suitable interaction device in an explicit way. For example, let us assume that a user is using a telephone in a living room. When s/he moves to a kitchen, s/he may use a speaker and a microphone in the kitchen. In this case, the user brings a token attached to the telephone, and put it into a base in the kitchen. Our system detects the event, and changes the interaction devices for the user.

However, implicit changes may be attractive for realizing pleasurable services. For example, if an environment detects that a user and his girl friend are together in a room, it is desirable to make the room's lighting strategy more romantic automatically. We believe that it is better to control the strategies for context-awareness should be customized in each application. The programming interface to control context-awareness should be clearly separated from other programming interface to make the structure of an application clear.

In the near future, designers for pervasive computing middleware should learn psychology, and we need to consider that the adaptation of software should not contradict our mental model. We believe that how to implement the real world model and the mental model in middleware infrastructures is an important research topic for building practical middleware for pervasive computing. For example, in our second middleware, if choosing a suitable interaction device is not consistent with a user' s mental model, the user will confuse which interaction device he should use. However, the implementation of real world model and mental model requires to represent ontologies in a standard way to access them from a variety of middleware for pervasive computing. We also need to consider social aspects and cultural aspects when designing applications interacting with the real world. For example, it is important to take into account trust and privacy in future pervasive computing environments, but we need to learn sociology and anthropology to know whether our understanding is enough or not. We believe that it is important to consider how to model psychological, social, and anthropological concepts into our programs to interact with the real world properly.

We also believe that the designers for pervasive computing middleware should know aesthetics to provide pleasurable services[3]. To develop pleasurable services, we may need to take into account emotion, peak experience, and unconsciousness to develop software. For example, our third middleware supports a mechanism to design pleasurable experiences by encoding preference rules in RF tags[11]. Our system infrastructure allows us to embed tags into various objects and places, and controls our experiences by changing the behavior of applications.

5 Research Agenda

We believe that it is important to consider the following three issues to achieve better integration between cyber and physical spaces.

The first issue is how to hide context-awareness from programmers when designing middleware infrastructures. Each application may have different criteria to consider how context should be abstracted. For example, when an application uses several interaction devices, some application programmers wish a middleware infrastructure to select the most suitable devices automatically, but other programmers want an application to select devices according to a user's intention. This means that a user's action triggers to change the currently used interaction devices, and application programs should extract the events to adapt their behavior. We believe that middleware infrastructures should not hide detailed context information when programmers want.

Currently, ubiquitous computing technologies are used to reduce a variety of complexities in our dairy lives. However, future ubiquitous computing applications should consider how to provide pleasurable services to users. We believe that it is important to consider a user's experiences to provide pleasurable services. When designing middleware infrastructures, we should consider how the middleware can support application programmers to design experiences explicitly.

As described in the previous section, we present that it is important to consider psychological aspects when designing middleware from a human factor's aspect. However, we also need to consider social aspects and cultural aspects when designing applications interacting with the real world. For example, it is important to take into account trust and privacy in future ubiquitous computing environments, but we need to learn sociology and anthropology to know whether our understanding is enough or not. We believe that it is important to consider how to model psychological, social, and anthropological concepts into our programs to interact with the real world properly.

6 Conclusion and Future Directions

This paper has described three middleware infrastructures that have developed in our project. We have also presented several experiences and future directions for building middleware for pervasive computing. We believe that there are new requirements to develop the middleware infrastructures for pervasive computing. Especially, we believe that it is important to take into account human factors to develop them.

Currently, we are developing a new middleware infrastructures for collaborating various appliances near a user[9]. In the system, we are conducting several experiments to consider the tradeoff between implicit changes and explicit changes to realize context-awareness. Also, we are working on developing another middleware to integrate various services provided in pervasive computing environments. In the middleware, we are considering to design programming interface to control context-awareness.

References

[1] W.Buxton, "Less is More(More or Less)", In Invisible Computing, P.J. Denning(Ed.), 2002.

[2] Edwards, K., Bellotti, V., Dey, A.K., Newman, M. "Stuck in the Middle: The Challenges of User-Centered Design and Evaluation for Middleware", In the Proceedings of CHI 2003, 2003.

[3] P.W.Jordan, "Designing Pleasurable Products", CTI, 2000.

[4] Tatsuo Nakajima, "Middleware Component Supporting Flexible User Interaction for Networked Home Appliances", ACM Computer Architecture News, December, 2001.

[5] T.Nakajima, D.Ueno, I.Satoh, H.Aizu, "A Virtual Overlay Network for Integrating Home Appliances", In the Proceedings of the 2nd International Symposium on Applications and the Internet, 2002.

[6] T.Nakajima, et. al., "Technology Challenges for Building Internet-Scale Ubiquitous Computing", In Proceedings of the 7th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, 2002.

[7] T.Nakajima, "Experiences with Building Middleware for Audio and Visual Networked Home Appliances on Commodity Software", In Proceedings of ACM Multimedia 2002, 2002.

[8] D.Ueno,, T.Nakajima, I.Satoh, K.Soejima, "Web-based Middleware for Home Entertainment", In Proceedings of International Conference on ASIEN'02, 2002.

[9] T.Nakajima, "Pervasive Servers: A Framework for Building a Society of Appliances", In Proceedings of the 1st International Conference on Appliance Design, 2003.

[10] . T.Nakajima. I.Satoh, "Personal Home Server: Enabling Personalized and Seamless Home Computing Environments", To be submitted, 2003.

[11] T.Nakajima, T. Akutagawa, "A Personalization Framework in a Personal Home Server: System Infrastructure for Designing Pleasurable Experiences", To be submitted, 2003.

[12] B.J.Pine II, J.H. Gilmore, "The Experience Economy", High Bridge Company, 1999.

[13] Eiji Tokunaga, Andrej van der Zee, Makoto Kurahashi, Masahiro Nemoto, Tatsuo Nakajima, "Object-Oriented Middleware Infrastructure for Distributed Augmented Reality", In Proceedings of IEEE International Symposium on Object-Oriented Real-Time Computing, 2003.

[14] K.Raatikainen, H.B.Christensen, T.Nakajima, "Applications Requirements for Middleware for Mobile and Pervasive Systems", ACM Mobile Computing and Communications Review, Vol.16, No.4, 2002.