Middleware Design and Human Factor

Tatsuo Nakajima
Department of Computer Science
Waseda University
tatsuo@dcl.info.waseda.ac.jp

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.

It is important to offer middleware infrastructures to hide a variety of complexities from application programmers to make easy to develop pervasive computing applications. 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.

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.

Such 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[1, 2, 3, 4, 5, 6, 7, 8]. 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

Our middleware infrastructure called MiRAGe 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.

3.3 Middleware for Interaction Devices

In the middleware infrastructure, 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.

3.4 Middleware for Home Computing

A personal home server 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.

4 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.

References

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

[2] 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.

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

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

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

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

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

[8] 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.