Softw are Infrastructure for Building Large-Scaled Smart Environments

Tatsuo Nakajima, Eiji Tokunaga, Hiro Ishikawa, Daiki Ueno, Mak oto Kurahashi,
Kunitoshi Iwasaki, Masahiro Nemoto, Andrej van der Zee
Department of Information and Computer Science
Waseda University
3-4-1 Okubo Shinjuku Tokyo 169-8555 JAPAN
tatsuo@dcl.info.w aseda.ac.jp

Abstract

This paper presents our ongoing project to develop software infrastructure for building large-scaled smart environments. In the future, our life will be augmented by embedding computers in our daily artefacts. However, without proper software infrastructure, it is very difficult to implemen t applications programs for the environments. Our software infrastructure provides a framework to build applications for augmenting our environments in an easy way. We describe an overview of our software infrastructure, and the current status of our project.

1 Introduction

Our daily life will be enhanced by embedding computers in our environments. Our environments will become smarter and more intelligent to help our dairy behavior, and our physical world is enhanced by using information in a cyber space. These environments are called *ubiquitous computing environments*[38, 24], and now there are many researches and commercial projects to try to realize the vision.

There are a lot of researches to develop user interface and applications to show the effectiveness of ubiquitous computing environments, but we found that it is not easy to develop applications for ubiquitous computing environments. Also, it is difficult to use various user interface devices in a seamless way for respective situations. Our experiences with build home computing systems[23, 26, 25, 36] show that providing high level abstraction is a key issue to build ubiquitous computing environments with a reasonable effort.

In this paper, we describe our ongoing project to develop software infrastructure for building large scaled smart environments. Our software infrastructure provides high level abstraction that is suitable for developing applications for smart environments. It supports to allow various services to be activated in a dynamic and context-aware way. Also, it provides various attractive features such as integrating physical spaces and cyber spaces, and processing continuous media for supporting augmented reality. On the top of the basic services, our software infrastructure enables us to manage various information from traditional documents to context information in a uniform way.

The remainder of this paper is structured as follows. In Section 2, we describe an overview of Internet-scale ubiquitous computing. Section 3 presents why middleware is important to build smart environments. In Section 4, we describe our middleware for building smart environmen ts, and Section 5 shows the current status of our project. Section 6 presents related work. Finally, we conclude the paper in Section 7.

2 Internet-Scale Ubiquitous Computing

The location "ubiquitous computing" was introduced by Weiser [38] to describe a scenario in which, literally, computing is everywhere. This should not be taken in the narrow-minded sense of "a computer on every desk", but in the rather subtler one of computers becoming em bedded in everyday objects and augmenting them with information processing capabilities. This em bedding would be discreet and unobtrusive: the computers would disappear from our perception, leaving us free to concentrate on the task at hand—unlike today, when a majority of users perceives computers as getting in the way of their work.

Part of this vision is already becoming a reality, in so far as most of us have already lost count of the number of gadgets we own that contain a microprocessor. Other parts of the vision, though, such as usability and universal interoperability, are still far away. Further developments are also envisaged: in the decade since the publication of W eiser's visionary article, the popularity of the W orld Wide Web has exploded, and it is easy to forecast that any ubiquitous computing infrastructure we will build in the future will perforce include global Internet connectivity. Moreover, the ongoing miniaturization of electronics, together with advances in power saving and wireless communication systems, has brought us to the stage where most of us are happy to *carry* computing and communication devices all the time. The most popular item for the general public is by far the cellular phone, but many also enjoy PDAs and digital music players. Mobility is now another important aspect of the ubiquitous computing scenario, and will be so even more as we evolve from gadgets we carry to gadgets we wear.

The next generation of home appliances is also going to be enhanced with communication and sensing capabilities. You can already buy CD players that connect to the Internet to look up the artist name and track titles for any CD you feed them; but this style of interaction will soon extend to non-electronic goods. The refrigerator will know that you are running out of milk and it will send a message to your phone when you are on the way to the shops—or it might reorder it directly by contacting the supplier through the Internet. The packaged food you wish to reheat will transmit the correct time and power settings to the microwave oven. And the washing machine will ask for confirmation before proceeding if it spots a white lace garment in a load of blue jeans.

There will be ubicomp-enhanced appliances not only in our homes but also in shared spaces such as offices, shops, restaurants and public transport vehicles such as trains. They will be connected to the Internet and might act as gateways (as well as direct interlocutors) for our mobile and wearable devices. We will be able to share and retrieve information wherever we are.

Devices and applications will become more responsive and friendly by personalizing their behaviour according to the preferences of the person who is using them. The capability of these systems to sense some aspects of their environment will make them adapt to the current situation. Location and environmental information, user identity, even user mood are some of the possible inputs to such systems [1, 29] that will allow applications to change their behaviour to fit the circumstances. New interaction techniques will become possible [32, 34, 18, 17, 13]; a variety of sensors will monitor our behavior, and improved models of the real world will provide better awareness of the context for the computing systems [14, 5, 7, 30, 4, 15].

In this paper we define Internet-scale ubiquitous computing as environments which embed computing, communication and sensing facilities and that are interconnected on a global scale, as shown in Figure 1. Any physical space can be augmented by embedding computers in it. Examples include the personal space (for body-worn devices), home spaces, vehicle spaces, office spaces, and university campus spaces. Public spaces such as airports, stations, bus stops, and public transport will also be globally connected, and different spaces will have different characteristics. For example, a personal space may contain a cellular phone, a PDA and an MP3 player that are connected by body networks. (Later these may be complemented by smart clothing.) Usually, appliances used in the personal space are small, battery-powered and wireless. On the other hand, in a home space, audio and video appliances are connected by wired high speed networks and are endowed with generous resources in terms of CPU, memory and energy.

In our projects, we are currently working on developping middleware for ubiquitous computing environments. We believe that our middleware will support various future ubiquitous computing applications such as smart space applications, entertainment applications such as robotics and game, and ad-hoc interpersonal communication applications. If there is no right middleware support, it is very hard to implement these applications.

We are currently organizing the following four projects:

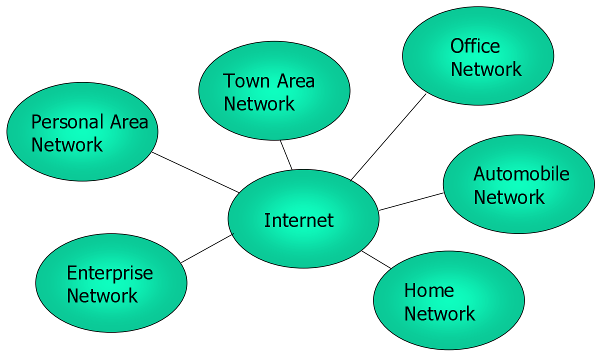

Figure 1: Internet-Scale Ubiquitous Computing

Autonomic Components, *Pervasive Servers*, *Universal Interaction*, and *Smart Materials*. The *autonomic component project* provides a component framework for ultra heterogeneous environments for building ultra portable applications. The framework makes it possible to adapt a component according to application requirements, platform requirements and environment requirements by supporting dependencies among components and assumptions of each component explicitly. In the *pervasive server project*, a large number of servers are embedded in our emvironments. Some servers contain sensors to provide information about the real world, and other servers allow us to control actual objects from programs. A mobile server that is wearable by each person integrates several pervasive servers near from the mobile server in an ad-hoc way. We are developping several applications on a mobile server to show the effecitiveness of our approach. The *universal interaction project* allows us to use any interaction devices to access various services. For example, a control panel can be shown at the nearest display from a user, and the panel can be navigated by various input devices such as a PDA, a cellular phone, and a game console. Also, the structure of the presentation is changed according to a user's situation. If a user can use both a large display and a small display, a video will be shown on the large display, but a control panel is appeared on the small display. Also, in the project, we are working on how several services are composed without noticing multiple services from a user. In the *smart marial* project, a world model is created by using various sensor technologies. Also, the project is developping a framework to accessing the world model. In the future, various sensors are contained in everyday materials. The goal of the project is to extract various information from everyday objects without a great effort.

3 Importance of Middleware

For building complex systems, abstraction is a very powerful tool to deal with the complexities. There are a lot of places where using abstraction is effective. For example, home computing applications require to access various home appliances such as televisions and video cassette recorders. The applications require to access these appliances without taking into account the differences among implemen tations to build portable applications. Also, ubiquitous computing applications require various new techniques such as service integration and context awareness. These techniques require good abstraction to build well structured applications.

Also, complex software usually adopts various techniques such as optimization and adaptation. Ad-hoc uses of the techniques make the structure of programs very unclear. For example, by inserting a lot of "if" statements to check the current situation, we can build context-aware applications. However, the program is very difficult to modify when the program needs to consider another situation. W e need good abstraction to support adaptive context-aware applications. Similarly, adopting aggressive optimization mak es programs less portable since the optimization may not be effective in another situation. W e believe that using better abstraction makes it possible to deal with optimization in a portable way.

Traditional system software provides low-level abstraction to control physical resources such as CPU, memory and devices. However, in smart environments, applications need to access various sensors devices and services. Also, it is important to abstract information from sensors and to support the composition of various services to build practical smart environments in an easy way. Therefore, middleware to support high level abstraction is necessary. Also, the middleware enables us to use various operating systems without modifying applications.

4 Middlew are for Building Smart En vironments

The section describes the design of our middleware for building smart environments. The middleware is assumed to run on commodit y software such as Java and Linux. We describe how commodit y software is used in our project after presenting an overview of our middleware. Then, we present two components in our middleware.

4.1 Overview of Our Arc hitecture

Our middleware uses commodity software to enhance its portability as much as possible. We have adopted CORBA as a comm unication infrastructure, and Java as an execution infrastructure. Also, we have adopted Linux as a underlying operating system.

Figure 2 shows the architecture of our system. Our middleware consists of the distributed component infrastructure and the content management infrastructure. The distributed component infrastructure is based on CORBA, and it allows us to comm unicate among components without taking into account the differences among operating systems and programming languages. On the distributed component infrastructure, we provides two types of components. The first type is a physical component that is used to control various appliances such as televisions and microwaves, or to access everyday artefacts such as cups and tables. The component is used to monitor various events in the real world, or to control everyday artefacts. The second type is a continuous media component that processes continuous media such as audio and video. The component is used to build augmented reality applications.

The content management infrastructure manages various types of information such as Web documen ts, context information, and information about components. The most important characteristic of the content management infrastructure is that information in cyber spaces can be enhanced by using various real world information captured by sensors. Lastly, applications provide a way to interact with users. Also, the content management infrastructure plays a role of a directory service for looking for a target component.

In our project, we are considering several types of applications. We are especially interested in smart space applications, entertainment applications such as games, and robotic applications. Our middleware integrates these applications to be cooperated in a seamless way.

4.2 Operating System Infrastructure

Complexity is a very serious problem to build ubiquitous computing environments. It is necessary to reduce the development cost because it is not realistic if everything should be implemen ted from scratch. We believe that using commodity software is very important to reduce the development cost because commodity software increases the portability of software.

Now, Linux can be used to build embedded systems, and a lot of software are available to build smart space applications in an easy way. Also, Java decreases the development cost dramatically due to the automatic memory management and the object-orientation. These software can be executed on both a standard personal computer and an embedded computer. Therefore, a program that is developed on a personal computer by using rich programming environmen ts is ported to an em bedded computer without modifying the program.

However, the current implementations of Linux and Java have are not suitable for building smart environments because the environments contain various types of computers that assume differen t requirements.

Currently, we are working on two projects to enhance Linux and Java. The first project is a customizable Java virtual machine. The virtual machine uses a component framework to make it more modular. In the project, we are currently working on which componen t framework is suitable for embedded systems. In the second project, we are enhancing security and resource management in Linux. Our current plan is to integrate Security-Enhanced Linux[19] and SILK[3] which is a version of Scout that runs as a kernel module in the Linux 2.4 kernel.

4.3 Distributed Componen t Infrastructure

Our middleware provides a uniform framework to access various componen ts. The framework supports to activate components in a dynamic way. The components are classified into components for physical objects and

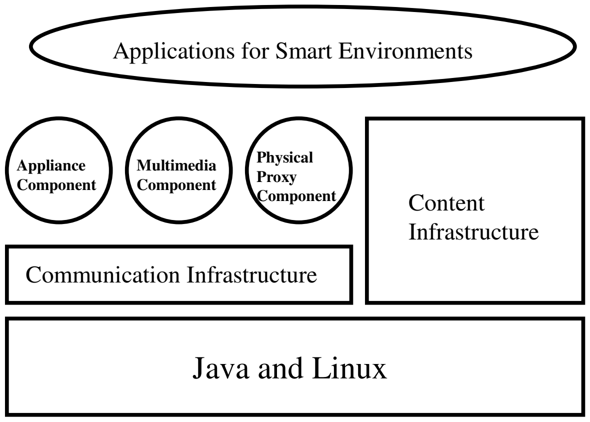

Figure 2: Overview of Our Architecture

continuous media components as shown in Figure 3. In this section, we explain how these components are implemented by using our framework.

The infrastructure provides a component abstraction to an upper layer. The component abstraction provides multiple interfaces, and each interface is identified by an each object reference. A client program is allowed to access a component in a location transparent way, but the program can specify a QOS value where a target component is activated in a context-aware way. In our framework, all physical objects are modeled by respective components, and a programmer has a unique view for all physical objects. We believe that the abstraction is very powerful to model our real world.

4.3.1 Dynamic Activation Managemen t

Our CORBA based framework supports components to be activated in a dynamic and context-aware way. Our system is extended to allow us to specify a QOS value to invoke a method on a target object similar to the proposal presented in [21]. Therefore, an application programmer describes where a component should be activated in an abstract way. For example, a user can specify to choose the nearest host to activate the service. Also, our system dynamically selects a host that can execute a component because the component may have a special requirement to be executed.

Figure 4 shows how our system activates a component in a dynamic way. First, a client acquires a object reference from an activation manager. The manager knows the current context in our environments by retrieving information from various sensors like RFID. When a request is delivered to the activation manager, a target host is selected according to the current context information, and an object reference for the target host's host activation manager is returned. Then, the request is forwarded to the host activation manager by using the location forward mechanism provided by the CORBA GIOP protocol. A target object is activated by the host activation manager, and its object reference is returned. Again, the location forward mechanism is used to forward the request to the target component.

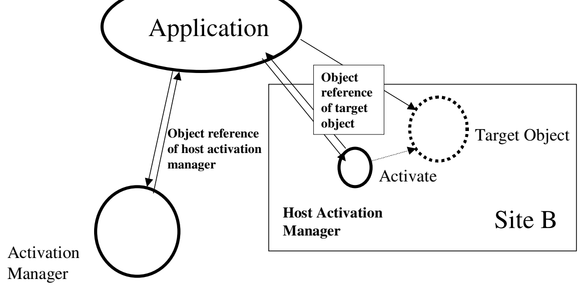

Figure 4: Dynamic Componen t Activation

Our client caches the object reference. Thus, if the effectiveness is expired, the object reference of the activation manager is used again to invoke a target component.

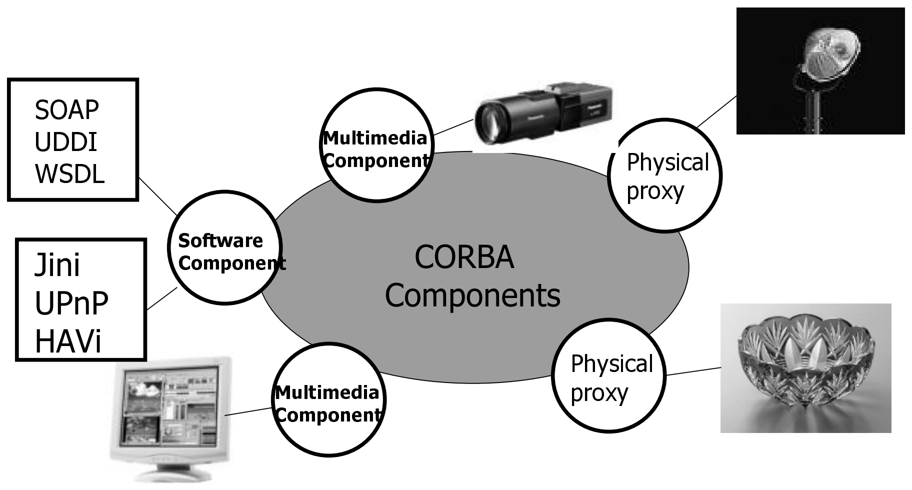

Figure 3: Distributed Infrastructure

4.3.2 Components for Physical Objects

The most important role of the components for physical objects is to control physical objects. We are considering two types of physical objects in our system. The first type of physical objects is an appliance such as televisions and microwaves. Recently, many appliances are connected to the Internet, and they allows us to be controlled by software. Actually, there are many protocols that are adopted by various appliances such as Jini, UPnP, HAVi, and SOAP. However, usually CORBA is not used to access appliances. Therefore, one of the role of the components is to convert protocols between the IIOP protocol and various protocols supported by appliances.

The second type of components is proxies of everyday artefacts. In our experiments, every physical object attaches a tag to identify itself. Also, a physical object may attach a Smart-Its device[35] or the object's material may be smart. Physical proxies may manage the location information of physical objects. Especially, it is important to monitor our location information to build context-aware applications.

The components for physical objects provide a standard interface for respective objects. These interface is defined by CORBA IDL, and can be accessed in a location transparent way. Also, the content management infrastructure works as a naming service because the infrastructure manages various information about respective components.

4.3.3 Continuous Media Management

Processing continuous media is very important to build smart environments. For example, the vision analysis is used to monitor the location information of artefacts surrounding us. Also, our real environments can be enhanced by using the augmented reality techniques. Continuous media components are used to help to build various types of ubiquitous computing applications.

Each continuous media component provides three interfaces. The first interface is a stream interface used to receive a stream data like audio and video streams. In our implementation, the interface does not defined by CORBA IDL, and each media element contains a header to identify the media type. The interface assumes to receive RTP packets, and all continuous media components need to provide the stream interface.

The second interface is an event interface that transmits events on a TCP connection. Each event contains a header to identify respective events. The interface is used to transmit information about each media element. The third interface is a control interface which is defined by CORBA IDL, but each component may have a different set of operations for respective continuous media components.

4.4 Content Management Infrastructure

Content management infrastructure manages various information such as various documents and information retrieved from our real world. Traditionally, various contents are stored in the Internet, but the contents have been created by a user. Therefore, most of information in our world could not be stored due to the cost to create the contents. However, if various information in the real world can be captured by various sensors that are embedded in everyday artefacts, it is not hard to collect most of information in the entire world. In the future, everyday artefacts will be made by small materials. The small materials contain very small invisible computers, and it

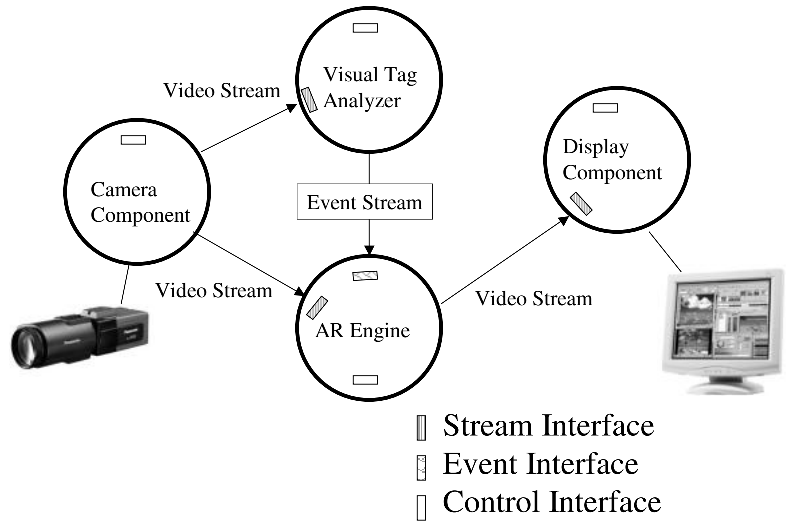

Figure 5: Continuous Media Management Components

is possible to retrieve various information by using the sensors. Also, smart materials may change the behavior of the artefacts using the collected information about the real world.

There are two important roles of the infrastructure. The first role is to provide a way to access various context information. The infrastructure stores various information about components described in the previous section, and a query about the real world is propagated to appropriate components to retrieve necessary information. Since it is very important to support context-aware applications in smart environments, an appropriate abstraction for context-awareness is a key issue to build practical smart environments. The second role is to provide a naming service for CORBA components. Our naming service is enhanced to return an object reference according to context information.

Figure 6 shows the structure of the content management infrastructure. Our system adopts a peer-to-peer infrastructure. The infrastructure provides efficient request routing, deterministic object location, and load balancing. In our system, various information is stored in various hosts on the peer-to-peer infrastructure. Our system also provides high level abstraction to retrieve the information stored in the content management infrastructure. The high level abstraction is based on distributed asynchronous collections[9] which provides object-oriented interface for publish/subscribe interaction. We believe object-oriented publish/subscribe interaction is a key abstraction to manage various information both in cyber spaces and physical spaces.

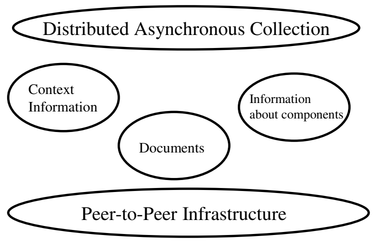
Distributed Asynchronous Collection
Context Information
Information about components
Documents
Peer-to-Peer Infrastructure

Figure 6: Content Managemen t Infrastructure

5 Current Status

Currently, our communication infrastructure has adopted omniORB2[16] as a CORBA runtime system. We are modifying omniORB2 to support dynamic activation and context-aware object invocation.

Also, we are working on a configurable Java virtual machine and enhanced Linux for building smart environments on commodity software.

The content managemen t infrastructure has adopted Pastry[33] which is developed by the Microsoft research labolatory, and we are working on adding asynchronous distributed collection abstraction on Pastry.

Lastly, we are also working on various components. Our current work focuses on continuous media components for building augmented reality applications. Also, some components convert CORBA protocol to SOAP protocol to access Web-based home appliances.

6 Related W ork

Recently, researches on building smart environments are a very active area in both academic research communities and industries. In [28], many projects about smart environments are introduced. In this section, we describe a couple of projects closely related to our project.

AT&T Labolatories, Cambridge has developed sentient computing environments that augment our real world by using various sensors[2]. Especially, ActiveBat is a wireless portable device that could be used for building context-aware computing. The device determines the location of each device in a fine grained resolution. The device is used to build location-aware applications.

Disappearing computing initiative[8] is an EU initiative to investigate methods and techniques for embedded computing in everyday artefacts. One such project of the initiative is Smart-Its[35], which investigates interconnected embedded technologies for smart artefacts with collective awareness.

The future computing environments research group at Geogia Tech is responsible for producing many of the more notable pieces of recent researches in the field of context awareness and ubiquitous computing[10]. The AwareHome project organized by the group focuses on to develop smart domestic environments.

The ambient research group has run several projects that investigate intelligent workspaces[11]. The i-Land project has produced several roomware devices: DynaWall - a large interactive smartboard, CommChair - an office chair with integrated computing and communication facilities, and InteracTable - a large table with an embedded table screen.

The Easy Living project[20] provides context aware computing using video tracking, recognition and sensor readings. Easy Living especially focuses on developing geometric mode of a room and taking reading from sensors embedded within.

A project by Philips to design *products, interfaces and lifestyles* of tomorrow[27]. The design and research has been focused on making familiar physical domestic objects from the kitchen through to the bedroom. Also, Philips Design has been augmenting everyday artefacts with an ambient intelligence. W earable devices, augmented toys and a living memory table have been produced as part of the project.

The interactive workspace project[37] concentrates on the integration of large screen displays and smaller personal computers in a meeting room environment. Novel interaction methods for interacting with the large screens in addition to information management and communication across devices are discussed.

There are several projects to aim to provide middleware for building smart environments. The middleware offers high level abstraction to make smart space applications easy.

one.world[12] developed at University of Washington is a software infrastructure for building pervasive applications. one.world provides high level abstraction to make it easy to implement pervasive applications. The system has been implemented in Java and it does not take into account building large-scale smart environments.

Gaia[6] is a distributed operating systems for building smart spaces. The system is based on CORBA. Also, their CORBA implementation can run on very small personal devices. The project is very similar to our system, but Gaia does not take into account the context-aware dynamic activation.

7 Conclusion and Future Work

The paper has described our ongoing project to develop middleware for building large scaled smart environments. Our middleware provides a couple of high level abstraction to build applications for smart environments without great efforts.

Currently, in our project, we have developed two more middleware components for realizing ubiquitous computing. The first middleware is home computing middleware that supports W eb-based protocols to control W eb-based home appliances[25, 26]. The middleware enables a user's request to forward to a target appliance in a context-aware way. The second middleware is for personal hand-held devices or wearable devices, which supports inter-personal communications. The middleware allows us to exchange information in an ad-hoc way.

In these middleware components, context-awareness is a key issue to realize ubiquitous computing environments. Especially, a technique to retrieve information from various sensors is a key issue in these middleware. W e need to study on a genetic support for context-awareness that is commonly available by our middleware components.

Also, it is important to develop a methodology to build software for ubiquitous computing environments. W e are currently working on a new paradigm to develop programs that take into account various issues in the real world such as timing constraints, resource constraints, context-awareness, and distribution.

References

[1] G.D. Abowd, E.D. Mynatt, "Charting Past, Present, and Future Research in Ubiquitous Computing". *ACM Transactions on Computer-Human Interaction*, 2000.

[2] AT&T Labolatories, Cambridge, "Sentient Computing", http://www.uk.research.att.com/.

[3] A. Bavier, T.Voigt, M.Wawrzomiak, L.Peterson, P.Gunningberg, "SILK: Scout Path in the Linux Kernel", Uppsale University, Technical Report 2002-009, 2002.

[4] B. Brumitt, J. Krumm, B. Meyers, S. Shafer, "Ubiquitous Computing and the Role of Geometry'. *IEEE Personal Communications*, August 2000.

[5] A.K. Dey, G. Abowd, D. Salber, "A Conceptual Framework and a Toolkit supporting the Rapid Prototyping of Context-Aware Applications". *Human-Computer Interaction (HCI) Journal* 16, 2001.

[6] Renato Cerqueira, Christopher K. Hess, Manuel Roman, Roy H. Campbell, "Gaia: A Development Infrastructure for Active Spaces", In Proceedings of Workshop on Application Models and Programming Tools for Ubiquitous Computing, 2001.

[7] K. Cheverst, N. Davies, K. Mitchell, A. Friday, "Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project". In "Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking", 2000.

[8] Disappearing Computer Initiative, http://www.disappearing-computer.net/.

[9] P. Eugster, R. Guerraoui and J. Sventek, "Distributed Asynchronous Collections: Abstractions for Publish/Subscribe Interaction", 14th European Conference on Object Oriented Programming (ECOOP 2000), 2000.

[10] Geogia Tech, Future Computing Environments, http://www.cc.gatech.edu/fce/.

[11] GMD-IPSI, Ambiante, http://www.darmstadt.de/ambiente/home.html.

[12] R.Grimm, et., al, "Programming for Pervasive Computing Environments, University Washington, Technical Reports, 2002.

[13] T. Hodes, R.H. Katz, "A Document-based Framework for Internet Application Control". In "Proceedings of the Second USENIX Symposium on Internet Technologies and Systems", 1999.

[14] A. Harter, A. Hopper, "A Distributed Location System for the Active Office". *IEEE Network Magazine*, 8(1), January 1994.

[15] A. Harter, A. Hopper, P. Steggles, A. Ward, P. Webster, "The Anatomy of a Context-Aware Application". In "Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking", 1999.

[16] omniORB2, http://www.uk.research.att.com/omni/.

[17] H. Ishii, B.Ullmer, "Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms". In "Proceedings of Conference on Human Factors in Computing Systems", 1997.

[18] N. Khotake, J. Rekimoto, Y. Anzai, "InfoStick: an interaction device for Inter-Appliance Computing". In "Proc. Workshop on Handheld and Ubiquitous Computing (HUC'99)", 1999.

[19] P.A. Loscocco, S.D. Smalley, "Meeting Critical Security Objectives with Security-Enhanced Linux", In the Proceedings of the 2001 Ottawa Linux Symposium, 2001.

[20] Microsoft Research, EasyLiving Project, http://www.research.microsoft.com/easyliving/

[21] T. Nakajima, "Practical Explicit Binding Interface for Supporting Multiple Transport Protocols in a CORBA system". In "Proceedings of International Conference on Network Protocols", 2000.

[22] T. Nakajima, "Towards Universal Software Substrate for Distributed Embedded Systems", In Proceedings of the International Workshop on Real-Time Dependable Systems, 2001.

[23] T.Nakajima, "System Software for Audio and Visual Networked Home Appliances on Commodity Operating Systems", In Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms, 2001.

[24] T. Nakajima, et. al., "Technology Challenges for Building Internet-Scale Ubiquitous Computing", In Proceedings of the Seventh IEEE International Workshop on Object-oriented Real-time Dependable Systems, 2002.

[25] 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.

[26] T.Nakajima and I.Satoh, "Naming Management in a Web-based Home Computing System", In Proceedings of the 20th IASTED International Multi-Conference on Applied Informatics, 2002.

[27] Philips, Home of the Near Future, Smart Connections, http://www.design.philips.com/smartconnections/.

[28] P.Phillips, "Smart Environment Research", In Proceedings of the Workshop on Ubiquitous Computing in Domestic Environments, 2001.

[29] R. Picard, *Affective Computing.* The MIT Press, 1997.

[30] T.-L. Pham, G. Schneider, S. Goose, "A Situated Computing Framework for Mobile and Ubiquitous Multimedia Access using Small Screen and Composite Devices". *ACM Multimedia*, 2000

[31] A.Reid, M.Flatt, L.Stoller, J.Lepreau, and E.Eide, "Knit: Component Composition for Systems Software", In Proceedings of the Fourth Symposium on Operating Systems Design and Implementation (OSDI 2000), 2000.

[32] J. Rekimoto, M. Saitoh, "Augmented Surfaces: A Spatially Continuous Workspace for Hybrid Computing Environments". In "Proceedings of CHI'99", 1999.

[33] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), 2001.

[34] I. Siio, T. Masui, K. Fukuchi, "Real-world Interaction using the FieldMouse". In "Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'99)", 1999.

[35] Smart-Its, http://www.smart-its.org/.

[36] K.Soejima, M.Matsuda, T.Iino, T.Hayashi, and T.Nakajima, "Building Audio and Visual Home Applications on Commodity Software", IEEE Transactions on Consumer Electronics, Vol.47, No.3, 2001.

[37] Stanford, Interactive Workspaces, http://graphics.stanford.edu/projects/iwork/.

[38] Mark Weiser, "The Computer for the 21st Century". *Scientific American* 265(3), 1991.