Service-Based Public Interaction Framework for Pervasive Computing

Tao Wang*, Yunxiang Ling, Guohua Zhang, and Huxiong Liao

C^4ISR Technology National Defense Science and Technology Key Lab, NUDT, Changsha, Hunan, China
U1987-2@163.com, yxling@tom.com, zghnudt@gmail.com, Liaohuxiong@163.com

Abstract. The idea of pervasive computing as people yearning for is not a long-standing version, and a unified interactive interface is a realistic demanding aspect of pervasive computing. To meet this requirement, a service-based framework called Public Interaction Framework (PIF) is presented in this paper. It focuses on allowing the applications that require complex interactions to run simultaneously on a public environment. PIF uses a device-based and task-oriented universal service called Public Interaction Service (PIS) to maintain the general support with multi-agent systems for the devices and applications registered to it. The importance of this work is providing an open, flexible framework that supporting the interactive equipment net in the pervasive computing environment, thus facilitating the application development.

Keywords: service-based, interaction framework, pervasive computing.

1 Introduction

This paper describes a public human-computer interaction framework called Public Interaction Framework (PIF) for the complex interactions in the pervasive computing environment.

In this paper we focus on the overall design and implementation of supporting complex interactions in the interaction systems. The PIF divides the whole interaction system into four spaces: user space, device space, service space and task space. And the main feature is the Public Interaction Service (PIS), the kernel component for the PIF to fit the requirements during the interaction process between human and computer.

In this paper we present the PIS as the typical service space for a public pervasive computing environment. We also discuss how it contributes towards satisfying the goals that to maintain the general support with multi-agent systems for the devices and applications registered to it.

2 Background

Mark Weiser coined the phrase "Ubiquitous Computing" around 1988, largely defined it and sketched its major concerns [1]. All models of Pervasive Computing

* This work is supported by NSFC 60875048.

share a vision of small, inexpensive, robust networked processing devices, distributed at all scales throughout everyday life and generally turned to distinctly common-place ends. And a lot of work has been done as efforts for the developers to achieve the goal of pervasive computing, such as the MIT Oxygen Project [2], the Microsoft Easy Living Project [3], the CMU Aura Project [4], the GIT Aware Home [5], etc.

Recently, numerous frameworks for pervasive computing have been proposed. Tianyin Xu et al. proposed the service discovery framework USDM-PerComp for pervasive computing environment, using a Web Service Server/Directory Server (WSS/DS) based two-level hierarchical topology [6]. And Jean-Yves TIGLI et al. presented extended SOA model for pervasive computing called Service Lightweight Component Architecture (SLCA), which presents various additional principles to meet completely pervasive software constraints [7]. With respect to the applications in the pervasive computing environment, Devdatta Kulkarni et al. presented a context-aware RBAC (CARBAC) model that focuses on the context-based access control requirements of such applications [8]. Dedicated to the development of pervasive computing applications, Wilfried Jouve et al. also presented a domain-specific Interface Definition Language (IDL) as well as its compiler [9]. Trevor Pering et al. had focus on pervasive collaboration with platform composition, and placed associated Composition Framework prototype [10].

3 Approach

As computers become cheaper, smaller and more powerful, they begin to appear in places where until recently we did not expect to find them. The idea of pervasive computing and smart environments is no longer a dream and has long become a serious area of research and soon this technology will start entering our everyday lives [11]. One of the major obstacles preventing this technology from spreading is the lack of interaction consistency. And the developers have presented some frameworks such as W3C multimodal interaction framework to resolve this problem, but there are still a lot of problems and challenges for the developers to face up if they decide to actualize them.

And as mentioned above, recent researches in pervasive computing field, including the frameworks, models and platform composition, have partly turned to services and components. Taking notice of that, though it is unpractical to support every interaction devices directly, the trend of modularization and service of the interaction technique, and the new progress of cross-platform technique, has brought us the possibility of public interaction service to support the complex interactions with the interactive equipment net in the pervasive computing environment.

Meanwhile, from the view of system engineering, the simple merge of the information from modals and devices cannot reach optimization. Thus, to enhance efficiency of the interactions, we should try to integrate the information from modals and devices, to divide the tasks between the devices and the software, with the full use of the rule of human cognizance such as relativity between information and actions. It has come to our notice that the human and computers have the different characteristics not only in the area of the interaction. So, follow the differences, let the human and computers do what they should do, do what they easily to do, and do what they adapt to do.

4 Public Interaction Framework

During the research and development of the intelligent room/space as a simple example of pervasive computing environment, Richard Hasha had found that a common distributed object platform was needed. After pointed the problem domain, he had also presented a prototypical object-network OS to support large numbers of objects and inter-object referencing [12].

Thus, the interactive equipments in the pervasive computing environment are deemed to make up a huge net with lots of subnets as shown in Fig.1. And each subnet includes lots of applications that requiring support for complex interactions. Our topic is to provide a unified interface/platform for the applications in the Interactive Equipment Nets (IENs).

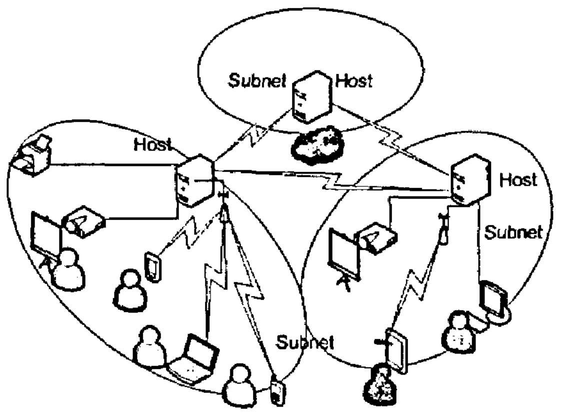

Fig. 1. The IENs of pervasive computing environment

In this paper, we describe the interaction system with the *Public Interaction Framework (PIF)* that divides the interaction system into four spaces: user space, device space, service space and task space as shown in Fig. 2.

User Space. The users are both the start points and end points in the whole interaction cycle. Corresponding to the intelligence level of the human cognizance, user space emphasizes the interaction capabilities of the users, especially the initiative that the computers and devices lack.

Device Space. Kinds of devices are the essential mediums for human-computer interaction. Device space emphasizes the interaction capabilities of the devices, as the devices produce the original interaction data, corresponding to the data level of the human cognizance.

Service Space. Running on the hosts. Services should be aware of devices and should be designed to explicitly interact with them. It is the kernel part of the framework. Thus the service space needs to provide a unified interface to all the devices that can be identified, and do its best efforts to support the applications based on it. The functions of service space have referred to the information and knowledge level of the human cognizance.

Task Space. To describe the requirements, task space has been called for. Task space has been designed to provide the applications with the resources limits and guidelines.

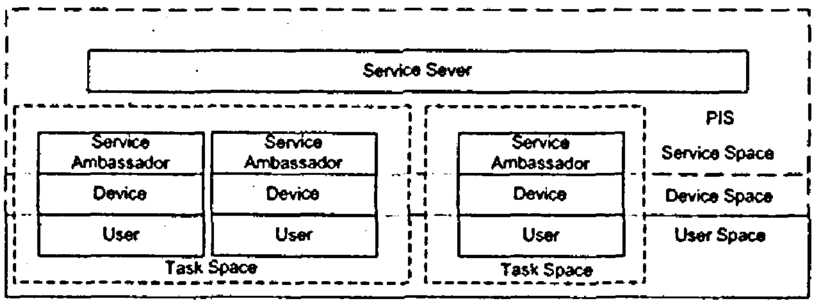

Fig. 2. Using four spaces to describe the interaction system

Given four spaces to describe interaction systems, we present the Public Interaction Framework (PIF) as shown in Fig. 3.

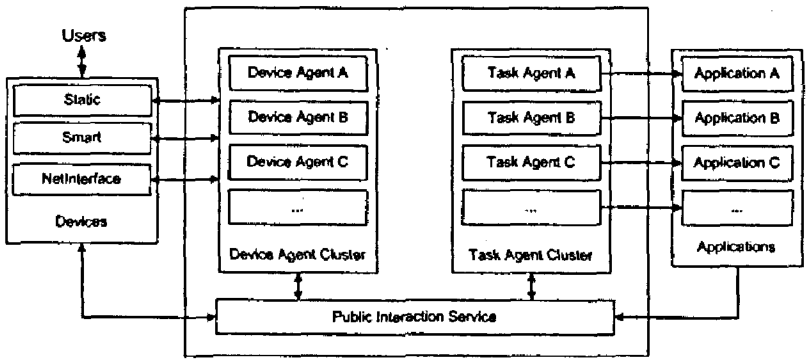

Fig. 3. The public interaction framework

To develop applications within the Public Interaction Framework, it is important to compartmentalize the functions properly, i.e. the developers should give each space the explicit requirements.

Obviously, it is not difficult for the developers to know what the devices that their applications need to support can do, but it is necessary to tell the services what the devices can produce. So, here the developers need to register the supported devices to the service providers in the service space. Meanwhile, the similar questions also exist in the task space. The developers should let the service space know what the given applications can afford to the users and devices. So, the developers also need to register the supported functions of the applications to the services.

The services work as the kernel of the framework with their main working flow as shown in Fig. 4. Actually, the services configuration and configure files are used for the initial/basic registrations mentioned above. After the registrations, the Central Control Unit (CCU) will control the service state and resources scheduling.

As the services are running, there are three main modules: device management, communication control, and task management. The device management module controls the device agents, the task management module controls the task agents, and the communication module controls the communications both of the local and networks. And with the help of communication control module, the CCU schedules the output device resources to satisfy the requests of applications.

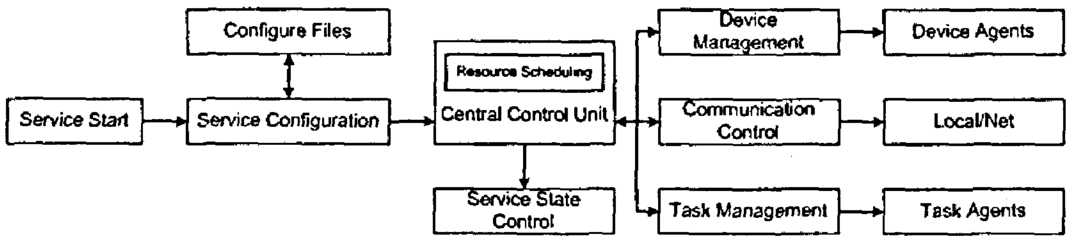

Fig. 4. The public interaction service

Device Agent

There are different device detection methods to identify different kinds of devices. To detect the static devices (i.e. the non-smart devices, such as keyboards, mouse, audio/video equipments, etc.), the service device detection process needs to query the registered devices table. To detect the smart devices (i.e. the smart cell phones, laptops, etc.), the service device detection process keeps listening. When the smart devices call to connect to the server, the process just detects them. After the PIS device detection process, the PIS device management module creates device agents for the identified devices. After this, the interaction data from the identified input devices will be captured by the input-device agents with their architectures as below shown in Fig. 5. Each input device will be binding to the proper device agent. And the input-device agents work following the processing steps, i.e. decodes the device data with sensor libraries, analysis modal data with engines, and detects the tasks, referring to the local state and communication with the PIS. For instance, Meng Xiangliang et al. had presented a new development model for multimodal interactive applications with unified access interface, and a collection of layered input primitives which is suitable for multimodal input interaction [13]. Meanwhile, the output devices are also controlled by the output-device agents as shown in Fig. 6. But the work that the output device agents do is simpler than the input-device agents; they just receive the directives from PIS and link to the device drivers for the application outputs. The following work will be transferred to the devices.

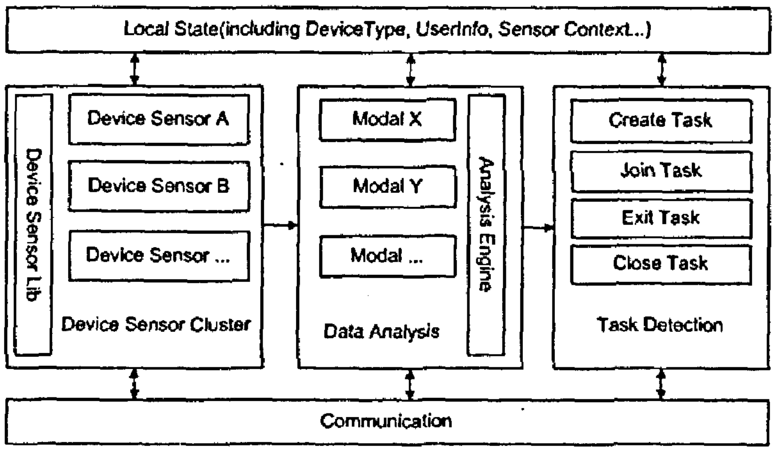

Fig. 5. The input-device agent architecture

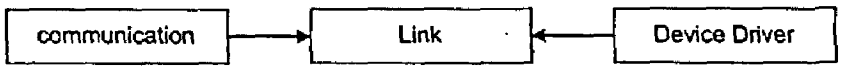

Fig. 6. The output-device agent architecture

Task Agent

The device agents trigger the executions of the task agents under the appropriate conditions, and task agents work with the given applications. Fig.7 shows the task agent architecture. Task agents get original instructions from the device agents though the PIS, process the context sensitive instruction assembly with the restrictions of conflict control strategy, and then collate the assembled command with local semantic library for the command linking with the local application interface.

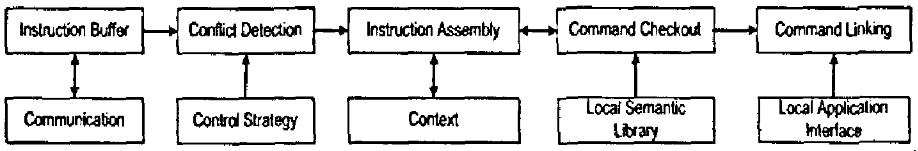

Fig. 7. The task agent architecture

The interaction has been once cycled after the process of task agent. And interaction cycles make up the whole process of the given applications. Thus, the framework has carried out the whole interaction system in the subnets of IENs for the pervasive computing.

5 Innovations and Related Works

Pervasive computing is not only a dream now; lots of papers propose various approaches to take into account specific information (for example, contextual information) into the applications in the pervasive computing environment. It is a classical way to address this topic though, in this paper we use the experience of other researches for reference and focus on existing approaches and techniques to support the interactions with the PIF.

Extending the topology of USDM-PerComp [6], we bring in the concept IENs as our pervasive computing environment, and in this paper we mainly work on the subnets of IENs.

Greatly influenced by HLA, we try to deal with dynamic device environment and dynamic application/software environment by using the PIS as RTI of PIF. To abate the computing load of the PIS CCU, agents are used as the ambassadors. The agents need to complete the data decoding, data analysis, and task detection with the help of interrelated techniques, for instance, context awareness that is very important for improving the intelligence of human-computer interaction [14].Though there are still lots of problems to be solved in the development of the agent based systems, Nicholas Hanssens et al have present concepts, technologies, and applications that they have developed to address the needs, ranging from low-level communication infrastructure to distributed applications with multi-modal interfaces, by taking an agent-base approach [15]. Stephen Peters has taken the existing Metaglue intelligent environment system and extend its capabilities to handle multiple users and spaces [16].

Taking the principles shared in SLCA into consideration [7], we notice that software-as-a-service (SAAS) technique is greatly called for, thus applications that linked by task agents, should provide their functions as services.

Standardization is one of the critical problems for the universal property and extendibility of the applications, so it is necessary for the PIF to work well. To reach the goal, a number of protocols need to be developed.

6 Conclusion

This paper has described our approach to allow the applications that require complex interactions to run simultaneously under a public framework in the pervasive computing environment. We have used the PIF to describe the whole interaction system and tried to provide the unified interfaces for the complex interactions with the help of PIS. The future work is to try to actualize and consummate the PIF in the future system development. And we first will focus on improving the PIS with lightweight ambassador components.

References

1. Weiser, M.: Computer of the 21st Century. J. Scientific American 265(3), 94–104 (1991)
2. MIT Project Oxygen, `http://oxygen.lcs.mit.edu`
3. Microsoft Research, EasyLiving Project, `http://www.research.microsoft.com/easyliving/`
4. Garlan, D., Siewiorek, D.P., Smailagic, A., Steenkiste, P.: Project Aura: Toward distraction-free pervasive computing. J. IEEE Pervasive Computing 1(2), 22–31 (2002)
5. Georgia Tech, Everyday Computing Project, `http://www.cc.gateçh.edu/fce/ecl/`
6. Xu, T., Ye, B., Kubo, M., Shinozaki, A., Lu, S.: A Gnutella Inspired Ubiquitous Service Discovery Framework for Pervasive Computing Environment. In: Proceedings of 8th IEEE International Conference on Computer and Information Technology, pp. 712–717. IEEE Press, New York (2008)
7. Tigli, J.-Y., Lavirotte, S., Rey, G., Hourdin, V., Riveill, M.: Lightweight Service Oriented Architecture for Pervasive Computing. J. International Journal of Computer Science Issues 4(1), 1–9 (2009)
8. Kulkarni, D., Tripathi, A.: Context-Aware Role-based Access Control in Pervasive Computing Systems. In: Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, pp. 113–122. ACM Press, New York (2008)
9. Jouve, W., Lancia, J., Palix, N., Consel, C., Lawall, J.: 6th IEEE Conference on Pervasive Computing and Communications, pp. 252–255. IEEE Press, New York (2008)
10. Pering, T., Want, R., Rosario, B., Sud, S., Lyons, K.: Intel Research Santa Clara: Enabling Pervasive Collaborationwith Platform Composition. In: Tokuda, H., Beigl, M., Friday, A., Brush, A.J.B., Tobe, Y. (eds.) Pervasive 2009. LNCS, vol. 5538, pp. 184–201. Springer, Heidelberg (2009)

11. Gajos, K.: A Knowledge-Based Resource Management System for the Intelligent Room. Thesis, Massachusetts Institute of Technology (2000)
12. Hasha, R.: Needed: A common distribute object platform. J. IEEE Intelligent Systems, 14–16 (1999)
13. Xiangliang, M., Yuanchun, S., Xin, Y.: Inputware: A Unified Access Interface for Multimodal Input Based on Layered Interaction Primitives. J. Harmonious Man-Machine Environment 2008, 242–247 (2008)
14. Abowd, G.D., Mynatt, E.D.: Charting Past, Present, and Future Research on Ubiquitous Computing. ACM Transactions on Computer-Human Interaction 7(1), 29–58 (2000)
15. Hanssens, N., Kulkarni, A., Tuchida, R., Horton, T.: Building Agent-Based Intelligent Workspaces. Technical report, MIT Artificial Intelligence Laboratory (2002)
16. Peters, S.: Infrastructure for Multi-User, Multi-Spatial Collaborative Environments. Technical report, MIT Artificial Intelligence Laboratory (2001)